# التعامل مع قواعد البيانات

# من خلال

# الفيجوال فوكس برو

صديق قيس عبد الرحمن الدوسري

جامعة الانبار- كلية الحاسوب

winron8@yahoo.com

## ١. مفهوم البرمجة غرضيه التوجه Object Oriented Programming

### أ . المقدمة

يعتبر OOP من أحدث التقنيات البرمجية في الوقت الحاضر، وقد ظهر هذا المفهوم في منتصف الثمانينات بعد ظهور البرمجة المهيكلة، وتعتمد الفكرة الأساسية في هذه التقنية على بناء كائنات (أو عناصر) لها خصائص محددة قابلة للتغير وتقوم بالاستجابة لأحداث وفق صيغ يمكن برمجتها بما يلائم التطبيق، وأهم محور في هذه الفكرة هو أن تكون هذه الكائنات قابلة للاستخدام أكثر من مرة ومن قبل أكثر من مستفيد مما يسهل عملية البرمجة من جهة ويوحد خصائص البرامج والتطبيقات من جهة أخرى.

### ب . الكائنات Objects

هي أصغر وحدة في OOP والتي يكون لها كيان مستقل ولها خصائصها التي يمكن تغييرها وصيغ استجابة محددة وقابلة للبرمجة للأحداث.

### ج . الصنوف Classes

هي عبارة عن التعريف المرجعي للقوالب التي يتم بناء الكائنات منها وهذه الصنوف تمتاز بصفات ثابتة تميزها عن بعضها البعض.

### د . الأحداث Events

هي عبارة عن فعاليات تحدث إما من قبل نظام التشغيل أو من قبل مستخدم الحاسبة، وهذا المفهوم هو من المفاهيم الأساسية في أنظمة التشغيل متعددة المهام (Multi Tasks Operating Systems) ومنها نظام التشغيل Windows، وكمثال على الأحداث هو عملية نقر زر الفأرة أو ضغط مفتاح معين في لوحة المفاتيح...الخ.

### هـ .الصيغ Methods

الصيغ أشبه ما تكون بالإجراءات في البرمجة المهيكلة (Procedures)، وهي عبارة عن الفعاليات التي تنفذ من قبل الحاسبة كاستجابة للأحداث التي تقع.

### و . الخصائص الأساسية للبرمجة بالكائنات الموجهة

هناك خصائص أساسية للبرمجة بالكائنات الموجهة وهي كالآتي:

#### اولا . التوارث Inheritance

هو عبارة عن إمكانية التصنيف الفرعي من صنف رئيسي وبناء ما يسمى Subclass من Class موجود مسبقا، وهذا Subclass يتوارث كل الصفات والخصائص والصيغ التي يمتلكها Class الأصلي (الأب)، لذلك فإن أي تغيير في صفات وخصائص الأب سوف ينعكس على الأبناء (Subclasses).

**ثانيا . التفرع Polymorphism**

هو عبارة عن إمكانية وجود صيغ (Methods) والتي لها نفس الاسم ولكن بمحتويات معالجة مختلفة، والصيغة التي سوف تستخدم تحدد في وقت التنفيذ من خلال صنف الكائن المستخدم.

**ثالثا . الدمج Encapsulation**

هو عبارة عن تضمين وإخفاء المعلومات عن كائن معين، مثل هيكل البيانات الداخلي للكائن. وعن طريق الدمج يتم عزل تعقيد عمل الكائن عن بقية التطبيق، كمثال إذا أردنا أن نكتب عبارة عنوان (Caption) على كائن معين فلا يهمنا في هذه الحالة معرفة كيفية خزن سلسلة الحروف (String) لهذه الكائن.

**ز . الكائنات المسيطرة (Control Objects)**

هي أصغر الوحدات البرمجية والتي لها خصائص وصيغ مثل Command Buttons، Text Box، ...الخ.

**ح . الكائنات الحاوية (Container Objects)**

هي عبارة عن كائنات يمكن أن تحتوي عل كائنات أخرى سواء كانت حاويات أو مسيطرات مثل Forms، Grid، ...الخ.

**ط . مفهوم الكائنات في VFP**

في VFP الكائنات الحاوية والمسيطرة هي الكائنات التي يمكن درجها في التطبيق ويمكن التعامل معها من خلال الخصائص (Properties)، الأحداث (Events)، الصيغ (Methods).

قد يبدو لنا أن الكائنات والصنوف هما نفس الشيء إلا أن هذا الاتجاه من المفهوم هو خطأ لأنهما لا يعنيان الشيء نفسه. الصنف (Class) يحتوي على معلومات عن كيفية ظهور وتصرف الكائن، أو بعبارة أخرى هو القالب الذي يصاغ منه الكائن. ولتقريب هذا المفهوم للأذهان نأخذ مثالا واقعيا فالمخطط الإلكتروني لجهاز الهاتف والتصميم الأساسي للجهاز من حيث أنه يمكن إجراء الاتصالات به ويمكن طلب الأرقام من خلاله يمكننا من اعتباره صنف Class (أي أنه صنف الأجهزة الذي يمكننا من الاتصال بالآخرين من خلاله) ويمكن اعتبار جهاز هاتف معين هو أحد الكائنات المنتمي لهذا الصنف. وبصورة عامة فإن الصنف يحدد الخصائص الأساسية للكائنات.

الكائن (Object) له عدد من الصفات (أو الخصائص) المحددة، على سبيل المثال جهاز الهاتف له لون وحجم ويأخذ موقع محدد في الغرفة.

الكائنات التي يتم إنشاؤها في VFP أيضا لها خصائصها المحددة والتي تتحدد من خلال الصنف الذي ينتمي له هذا الكائن وهذه الخصائص يمكن تغييرها وإعطائها قيم معينة إما أثناء وقت التصميم (Design Time) أو إثناء وقت التنفيذ (Run Time). والجدول (١-١) يبين عدد من الخصائص التي قد تمتلكها كائنات VFP.

ومن جهة أخرى الكائنات يمكن أن تستجيب لأحداث محددة والتي هي عبارة عن فعاليات تتولد من قبل نظام التشغيل أو من قبل المستخدم والجدول (٢-١) يبين عدد من الأحداث في VFP.

وكاستجابة للأحداث يوجد هناك ما يسمى بالصيغ (Methods) وهي عبارة عن عبارات برمجية معينة تنفذ عند وقوع الحدث، وهي شبيهة إلى حد ما بالإجراءات (Procedures) في FP العادية إلا أنها تختلف من حيث أنها مرتبطة بالكائن وتختلف أيضا في طريقة استدعائها، وعلى العموم فإن الأحداث (Events) في VFP تكون محددة ولا يمكن إضافة أحداث جديدة إلى التطبيق، إلا أنه من الممكن إنشاء صيغ (Methods) جديدة. والجدول (٣-١) يبين عدد من الصيغ في VFP.

| الخاصية | الوصف |
|---|---|
| Caption | العبارة التي تظهر على الكائن |
| Forecolor | لون الكائن |
| Backcolor | لون الخلفية للكائن |
| Left | إزاحة الكائن عن يسار الشاشة |
| Top | إزاحة الكائن عن أعلى الشاشة |
| Visible | خاصية منطقية ( True أو False) يحدد ظهور الكائن في الشاشة |
| Enabled | خاصية منطقية ( True أو False) يحدد هل الكائن فعال أو غير فعال |
| Font | أسم البنط لحروف الكائن |

**الجدول (١-١) بعض الخصائص لكائنات VFP**

| الحدث | الوصف |
|---|---|
| Click | نقر زر الفأرة الأيسر |
| Gotfocus | المستخدم يختار الكائن إما بالنقر عليه أو عن طريق Tab |
| Lostfocus | المستخدم يختار كائن آخر غير الكائن الحالي |

**الجدول (٢-١) بعض الأحداث في VFP**

| الصيغة | الوصف |
|---|---|
| Refresh | إعادة تنشيط الكائن لعكس التغيرات التي قد تكون حدثت |
| Setfocus | نقل الاختيار إلى كائن معين |

**الجدول (٣-١) بعض الصيغ في VFP**

## ٢ . تصميم قاعدة البيانات في VFP

في VFP يتم استخدام قاعدة البيانات DATABASE لغرض تنظيم وربط الجـــداول Tables والمنـــاظر Views. وقاعدة البيانات في VFP يقدم معمارية ترتيب البيانات بالإضافة إلى ميزات أخرى تتضمن قواعد إدخـــال وتحـــديث القيود في الجداول، إعطاء الحقول قيم أولية ثابتة، التحقق من قيم القيود عند الإدخال أو التحديث (Triggers)، إنشاء الإجراءات (Procedures) لكل جدول، إنشاء علاقات ثابتة ورصينة بين جداول قاعدة البيانات دون الحاجة إلى بناء هذه العلاقات أثناء التنفيذ، بالإضافة إلى إمكانية إنشاء Views محلية أو بعيدة للبيانات والجداول ضمن القاعدة.

### أ . إنشاء قاعدة البيانات

بعد تصميم قاعدة البيانات، من الممكن إنشاء القاعدة إما عن طريق واجهـــة المـــستخدم (User Interface) أو عـــن طريق البرمجة (كتابة إيعاز الإنشاء)، وبعد ذلك يمكن تحديث محتويات قاعدة البيانـــــات بالاســـــتفادة من خاصـــية قاموس البيانـــات (Data Dictionary) الموجودة في VFP.

عند إنشاء قاعدة البيانات يتم تجميع الجداول في حاوية واحدة للاستفادة من خاصية قاموس البيانات التي يمكـــن مـــن خلالها إنشاء وتعريف ما يلي:-

اولا . المفاتيح الأولية والمفاتيح المرشحة.

ثانيا . العلاقات الثابتة بين الجداول.

ثالثا . الأسماء الطويلة للجداول والحقول.

رابعا . عبارة العناوين (Captions) للحقول لإظهارها في نافذة Browse وفي Grids.

خامسا . القيم البديهية لحقول.

سادسا . الصنف البديهي للحقول في شاشات النماذج.

سابعا . قواعد التحقق للحقول والقيود.

ثامنا . خزن الإجراءات الخاصة بقاعدة البيانات.

تاسعا . الهوامش والتعليقات للجداول والحقول.

و لإنشاء قاعدة بيانات جديدة :

نختار من File الاختيار New ومن ثم نختار Database ونعطيه اسما فتظهر لنا الحاوية.

نستخدم إيعاز **CREATE DATABASE**.

كمثال:-

create database *sample(or any other name)*

ومن ثم نكتب

modify data [*sample*] *(or any other name)*

فتظهر لنا النافذة التالية:

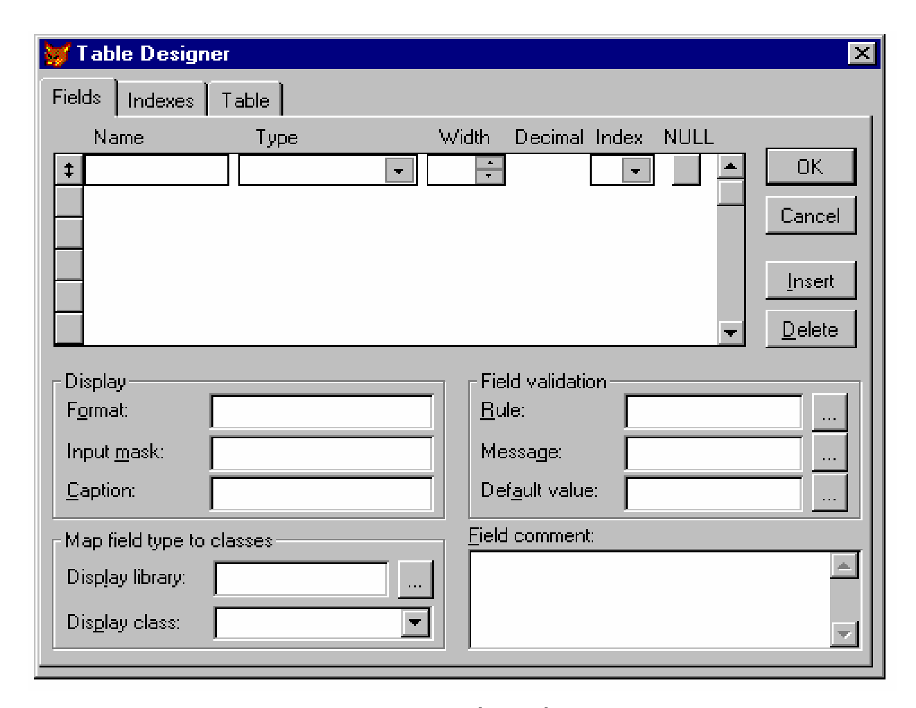



**الشكل (٢ - ١) واجهة تصميم قاعدة البيانات Database Designer**

**ب . التعامل مع الجداول**

**اولا . إنشاء جدول جديد في القاعدة**

يمكن إنشاء جدول جديد في القاعدة الحالية إما عن طريق الضغط على Icon الإنشاء أو عن طرق النقر على الــزر الأيمن للفأرة واختيار New Table من قائمة الاختيارات التي سوف تظهر، وبصورة عامة فــإن شاشــة تــصميم الجداول سوف تظهر كالآتي:-




**الشكل (٢ - ٢)**

**ثانيا . إضافة جدول موجود مسبقا إلى قاعدة البيانات**

عند اختيار Icon الخاص بهذه الفعالية يظهر صندوق محادثة Dialog Box الخاص بنظام التشغيل لاختيار الجدول الموجود في مكان معين من القرص ويتم إضافته إلى قاعدة البيانات.

**ثالثا . تسمية الجدول**

عند استخدام إيعاز Create Table يتم إدراج أسم الجدول ضمن الإيعاز لغرض خزن الجدول بهذا الاسم. أسم الجدول يمكن أن يتألف من أحرف، أرقام، وعلامة التسجيل التحتية (underscores) ويجب أن تبتدأ بحرف أو underscore.

وفي حالة كون الجدول ضمن قاعدة البيانات فمن الممكن إعطاء الجدول أسم طويل يصل إلى حد ١٢٨ رمز.

**رابعا . تغيير أسم الجدول**

يمكن تغيير أسم الجدول من خلال واجهة تصميم قاعدة البيانات فقط (الشكل ٢-١) وذلك لأن المطلوب تغييره هو الاسم الطويل للجدول، ولتغيير أسم الجدول في قاعدة البيانات نتبع الخطوات التالية:-

* في واجهة تصميم قاعدة البيانات (Database Designer) نختار الجدول المطلوب تغيير اسمه.

* من قائمة Database نختار Modify، أو عن طريق نقر الزر الأيمن للفأرة على الجدول المطلوب لاستعراض قائمة العمليات على الجدول ونختار Modify.

* في شاشة مصمم الجدول Table Designer نطبع أسم الجدول في حقل Table Name ضمن Table tab. (الشكل ٢-٢)

**خامسا . حذف جدول من قاعدة البيانات**

هناك طريقتين لحذف الجدول من قاعدة البيانات:-

*حذف الجدول من القاعدة دون حذفه بصورة نهائية من القرص ويسمى remove.

*حذف الجدول من القاعدة ومن القرص ويسمى delete.

ولحذف الجدول نتبع الخطوات التالية:-

*من database designer نختار الجدول المراد حذفه، ومن قائمة database نختار delete ضمن الاختيارات الثلاث التي سوف تظهر وهي remove, delete, cancel. وفي هذه الحالة سوف يحذف الجدول من القاعدة ومن القرص، أما عند اختيار remove فإن الجدول يحذف من القاعدة فقط.

**أو**

* عن طريق نقر الزر الأيمن للفأرة (right click) على الجدول المراد حذفه فتظهر نفس الشاشة السابقة ذات الاختيارات الثلاث.

## سادسا . تسمية الحقول

قواعد تسمية الحقول هي نفس قواعد تسمية الجداول. هذه الحقول يصل طولها إلى عشرة رموز في حالة free table ويصل إلى ١٢٨ رمز في حالة كون الجدول ضمن قاعدة البيانات.

ولغرض تسمية الحقل يتم إدخال أسم الحقل في مربع Name في table designer (الشكل ٢-٢).

## سابعا . اختيار نوع البيانات للحقول

عند إنشاء أي حقل يجب إدخال نوع البيانات الخاص بهذا الحقل والذي يحدد الخواص التالية:-

* طبيعة القيم التي ستدخل في هذا الحقل (character, numeric،...etc).

* حجم الحقل (مساحة الخزن).

* طبيعة العمليات التي يمكن إجرائها على الحقل، فمثلا يمكن جمع حقل numeric أو currency إلا إن هذا غير ممكن بالنسبة memo أو General.

* إمكانية عمل فهرسة (Index) أو ترتيب (Sort) على الحقل تتدخل فيه أيضا نوعية بيانات هذا الحقل، فمثلا يمكن فهرسة حقل من نوع character ولا يمكن عمل فهرسة على حقل memo.

والجدول التالي يبين أنواع البيانات في VFP:-

| نوع البيانات | الوصف | الحجم | المدى |
|---|---|---|---|
| **Character** | Any text | ١byte per character to 254 | Any characters |
| **Currency** | Monetary amounts | ٨bytes | ٩٢٢٣٣٧٢٠٣٦٨٥٤٧٧,٥٨٠٧ -to 922337203685477.5807 |
| **Date** | Chronological data consisting of month, year, and day | ٨bytes | When using strict date formats ،{٠١-٠١-٠٠٠١^} ، January 1st, 1 A.D to ،{٣١-١٢-٩٩٩٩^} December 31st, 9999 A.D. |
| **DateTime** | Chronological data consisting of month, year, day, and time | ٨bytes | When using strict date formats ،{٠١-٠١-٠٠٠١^} ، January 1st, 1 A.D to ،{٣١-١٢-٩٩٩٩^} December 31st, 9999 A.D., plus 00:00:00 a.m. to 11:59:59 p.m. |
| **Logical** | Boolean value of true or false | ١byte | True (.T.) or False (.F(. |
| **Numeric** | Integers or fractions | ٨bytes in memory؛ ١to 20 bytes in table | ٩٩٩٩٩٩٩٩٩٩٩. -E+19 to .9999999999E+20 |
| **Double** | A double-precision floating-point number | ٨bytes | ٤,٩٤٠٦٥٦٤٥٨٤١٢٤٧-/+E-324 to +/- 8.9884656743115E307 |
| **Float** | Same as Numeric | ٨bytes in memory; 1 to 20 bytes in table | ٩٩٩٩٩٩٩٩٩٩٩. -E+19 to .9999999999E+20 |
| **General** | Reference to an OLE object | ٤bytes in table | Limited by available memory |
| **Integer** | Integer values | ٤bytes | ٢١٤٧٤٨٣٦٤٧-to 2147483647 |

| | | | |
|---|---|---|---|
| **Memo** | Reference to a block of data | ٤bytes in table | Limited by available memory |
| **Character (Binary(** | Any character data you want to maintain without change across code pages | ١byte per character to 254 | Any characters |
| **Memo (Binary(** | Any memo field data you want to maintain without change across code pages | ٤bytes in table | Limited by available memory |

## ثامنا . إضافة فهرسة منتظمة Adding Regular Index

عند إضافة أي حقل، يمكن إضافة فهرسة منتظمة regular index عليه من خلال تأشير المربع المسمى Index، وهذه الفهرسة تكون إما تصاعدية أو تنازلية حسب مؤشر السهم. هذه الفهرسة سوف تضاف إلى فهرسة الجدول ويمكن تحديثه من خلال الضغط على Index Tab في مصمم الجدول Table Designer.

## تاسعا . استخدام قيم اللاشيء Null Values

عند إنشاء جدول جديد، يمكن تحديد حقل واحد أو أكثر ليتقبل القيم اللاشيء. عند استخدام Null values فإننا نثبت الحقيقة التي مفادها أن المعلومات المخزونة في هذا الحقل لقيود معينة غير محددة في الوقت الحاضر (أو عند إدخال القيد). وبدلا من استخدام صفر أو فراغ والتي قد تعني شيء ذو معنى يستخدم null في قيمة هذا الحقل إلى أن تتاح المعلومة ذات المعنى.
ولغرض إدخال Null لحقل معين، يتم تأشير مربع Null لهذا الحقل بنقر زر الفأرة الأيسر عليه (تظهر علامة صح، وعند الضغط عليه مرة ثانية يختفي التأشير). ويجب ملاحظة أن الإيعاز SET NULL هو في حالة ON لكي يصبح Null Values فعالا في جميع الجداول.

## عاشرا . إضافة التعليقات لحقول الجدول

يمكن إضافة التعليقات لأي حقل في الجدول وذلك لجعل فهم الجدول سهلا بالنسبة لمصمم القاعدة أو بالنسبة لمطوري برامج القاعدة ومدير قاعدة البيانات، ويتم إضافة التعليقات من خلال مربع النص المسمى Field comment والذي يقع أسفل يمين شاشة مصمم الجدول Table Designer.

## احد عشر . إنشاء القيم البديهية للحقول Default Field Values

عندما نريد أن يقوم VFP بإملاء قيم الحقل بصورة أوتوماتيكية عند إضافة قيد جديد إلى الجدول فإننا نستخدم Default Value، القيمة التي تدخل في الجدول تبقى فيه إلى أن يتم تغييره. (يمكن تحديد Default Value لكل أنواع البيانات ما عدا General). ولتحديد Default Value لحقل معين يتم إدخال هذه القيمة في مربع Default Value في المساحة المسماة Field Validation. قيمة Default Values تسرع من عملية إدخال البيانات في التطبيق، حيث يتيح للمستخدم أن يعبر الحقل دون إدخال قيمة فيه (ما لم يريد أن يخزن فيه قيمة مختلفة)، وهذه الحالة تكون مفيدة جدا خاصة إذا كانت هناك قواعد إدخال على مستوى القيد أو الحقل وهو ما

يسمى field-level validation rules و validation rules record-level. قيم Default Values يمكن أن تكون أرقام أو تعبيرات حرفية أو حسابية أو منطقية وقد تكون عبارة عن دالة معرفة من المستخدم UDF.

**اثنا عشر . تحديد قناع الإدخال Input Mask**

عند استخدام input mask فإننا نعرف صيغة الإدخال للقيمة البيانية (يتم ذلك من خلال تحديد الفواصل والفوارز والفراغات في صيغة الإدخال)، ويتم هذا عن طريق إدخال الصيغة في مربع Input Mask في المساحة المسماة Display.

**ثلاثةعشر . السيطرة على أسلوب عرض الحقل**

هناك عدد من الخصائص التي تمكننا من السيطرة على أسلوب عرض الحقل سواء في Forms، Browse Window، أو في التقارير. فيمكن تحديد صيغة العرض (Display Format)، عنوان العرض (Field Caption)، وصنف الحقل (Default Class).

في المساحة المسماة Display يمكن تحديد صيغة العرض من خلال مربع Format وهي الصيغة التي تظهر فيها قيمة الحقل في جميع صيغ الإخراج في VFP.

ويمكن أيضا إدخال عنوان العرض من مربع Caption في نفس المساحة، هذا العنوان سوف يظهر في Forms، Browse، وفي التقارير بدلا من أسم الحقل في الجدول.

ولاختصار الوقت عند تصميم النماذج (Forms)، يمكن تحديد Default Class لكل حقل، ففي كل مرة يتم فيها إنشاء نموذج لهذا الجدول فإن Default Control التابع إلى Default Class المختار سوف يظهر في النموذج.

**اربعةعشر . قواعد التحقق Validation Rules**

يمكن استخدام قواعد التحقق بأسلوبين وهما:- field-level و record-level، وذلك للسيطرة على البيانات الداخلة في جداول قاعدة البيانات. تقوم هذه القواعد بالتحقق من القيمة المدخلة مع الشروط التي تتضمنها قواعد التحقق، فإذا كانت مستوفية للشروط فإن البيانات سوف تقبل وإلا فإنها سوف ترفض. وللعلم فإن قواعد التحقق تكون متوفرة فقط في الجداول المقترنة ضمن قاعدة بيانات وغير متوفرة في الجداول الحرة. قواعد التحقق تختصر عملية كتابة هذه القواعد في اختيار VALID في النماذج أو في المقاطع البرمجية في إيعازات البرنامج.

كما ويمكن استخدام فهرسة الحقول من نوع Primary و Candidate لمنع تكرار قيمة لحقل معين، وكذلك استخدام Triggers لتوجيه ما يسمى بوجوب التكاملية (Referential Integrity) للقيام بفعاليات معينة عند تغيير البيانات في القاعدة، كل هذا يمكن إدراجه ضمن قواعد التحقق لجداول قاعدة البيانات.

**خمسةعشر . معرفة متى توجه المحددات**

يتم اختيار محددات قاعدة البيانات بالاعتماد على مستوى التحقق، وكذلك على العملية التي تجعل المحدد فعالا. يبين الجدول التالي محددات تحقق البيانات في VFP:-

| Enforcement Mechanism<br>ميكانيكية التوجيه | Level<br>المستوى | Activated<br>الفعالية |
|---|---|---|
| NULL validation | Field or column | When you move out of the field/column in a browse, or when the field value changes during an INSERT or REPLACE. |
| Field-level rules | Field or column | When you move out of the field/column in a browse, or when the field value changes during an INSERT or REPLACE. |
| Record-level rules | Record | When the record update occurs. |
| Candidate/primary index | Record | When the record update occurs. |
| VALID clause | Form | When you move off the record. |
| Triggers | Table | When table values change during an INSERT, UPDATE, or DELETE event. |

**ستةعشر . تحديد القيم لحقل معين في الجدول**

عندما نريد السيطرة على البيانات التي يتم إدخالها من قبل المستفيد، وإذا كنا نريد أن نتحقق من قيمة الحقل بغض النظر عن القيم في الحقول الأخرى نستخدم field-level validation.

ولإنشاء Field-level validation ندخل التعبير الخاص في مربع Rule في المساحة المسماة Field validation area الموجودة في Table Designer.

ولإضافة التعبير الذي يظهر أثناء حدوث خطأ معين ندخل التعبير في مربع Message في المساحة Field validation.

VFP يتحقق من Field-level rules عند تغيير قيمة الحقل، وهو على العكس من Triggers فهو يتم التحقق منه حتى إذا كنا نستخدم Buffered Data.

وبالنسبة إلى Record-level validation فهو يسيطر على نوع البيانات لقيد واحد، وذلك عن طريق عمل مقارنة بين قيم حقلين أو أكثر، فمثلا إذا أردنا أن نتحقق من كون قيمة أحد الحقول هو أكبر دائما من قيمة حقل آخر. ولإنشاء هذا النوع من التحقق ندخل التعبير وعبارة الخطأ في مربعي Rule و Message في Table Designer في اختيار Table tab (الشكل ٢-٢).

**تنبيـــــه**

*لا تضع إيعاز أو Function في Validation rules من التي تسبب تحريك مؤشر القيد ( Record Pointer) في مساحة العمل الحالية (Current Work Area)، مثل إيعازات Seek، Locate، Skip، Append) والتي قد تسبب حدوث نبضات متداخلة recursive triggers مما يؤدي إلى حدوث خطأ.*

**سبعةعشر . استخدام Triggers**

يعرف Triggers على أنه التعبير الذي يرتبط بالجدول في قاعدة البيانات، ويتم استدعاء هـذا التعبيـر كلمـا حدث تغيير في بيانات حقول الجدول. يتم إنشاء Triggers وخزنها بشكل خاصية من خواص الجدول داخـل قاعدة البيانات، لذلك عند حذف الجدول من القاعدة تنتهي Triggers الخاصة بهذا الجـدول بـصورة نهائيـة. ويتميز Triggers عن باقي قواعد التحقق في أنه لا يستدعى عندما تكون البيانات في حالة Buffered.

**ثمانيةعشر . إنشاء Triggers**

لإنشاء Triggers يتم إدخال التعبير الخاص به trigger expression من خلال شاشة مصمم الجدول Table Designer من اختيار Table tab (الشكل ٢-٢) من خلال المربعـات Insert trigger، Update trigger، و Delete trigger.

فمثلا إذا أردنا إنشاء trigger في نظام سيطرة على الخزين يقوم بإضافة حقل في جدول المواد (ولـيكن أسـم الجدول Products) وهذا الحقل أسمه (ليكن reorder_amount) يتم خزن كمية المادة المطلوب تجهيزها فيه عند وصول المادة إلى مستوى إعادة الطلب (وهو أيضا حقل ليكن أسمه reorder_level) ويتم وضع حقلـي رقم المادة (product_id) وحقل كمية إعادة الطلب (reorder_amount) في جـدول جديـد (لـيكن أسـمه reorder) للاستفادة من هذا الجدول في استخراج التقارير الخاصة بإعادة الطلب. هذا trigger يتم اسـتدعائه عند تثبيت صرف كمية من المادة للتحقق من وصول المادة إلى مستوى إعادة الطلب، هذا trigger قد يكـون بالصيغة التالية:-

```
PROCEDURE updProductsTrigger
  IF (units_in_stock+units_on_order) <= reorder_level
  INSERT INTO Reorder VALUES(Products.product_id, ;
   Products.reorder_amount)
  ENDIF
ENDPROC
```

ولحذف أو تحديث trigger موجـود يتم الدخـــول إلى trigger عن طريـق مربعـي delete trigger و update trigger في شاشة Table Designer.

## ٣ . التعامل مع قاعدة البيانات في VFP

قاعدة البيانات توفر بيئة عمل يتم من خلالها إنشاء وخزن مجموعة من الجداول، ويـــوفر أيـــضا إمكانيـــة إنـــشاء العلاقـــات (relationships) بين الجداول، ويوفر بناء مجموعة من قواعد التحقق والسيطرة على البيانات. تخزن قاعدة البيانـــات فـــي ملف له امتداد من نوع ( .dbc).

### أ . فتح قاعدة بيانات

يتم فتح قاعدة البيانات إما عن طريق اختيار قائمة File ومن ثم open واختيار قاعدة البيانات المراد فتحهـــا، أو عن طريق إيعاز open database كالآتي:-

open database (database name)

### ب . تحديث محتويات قاعدة البيانات

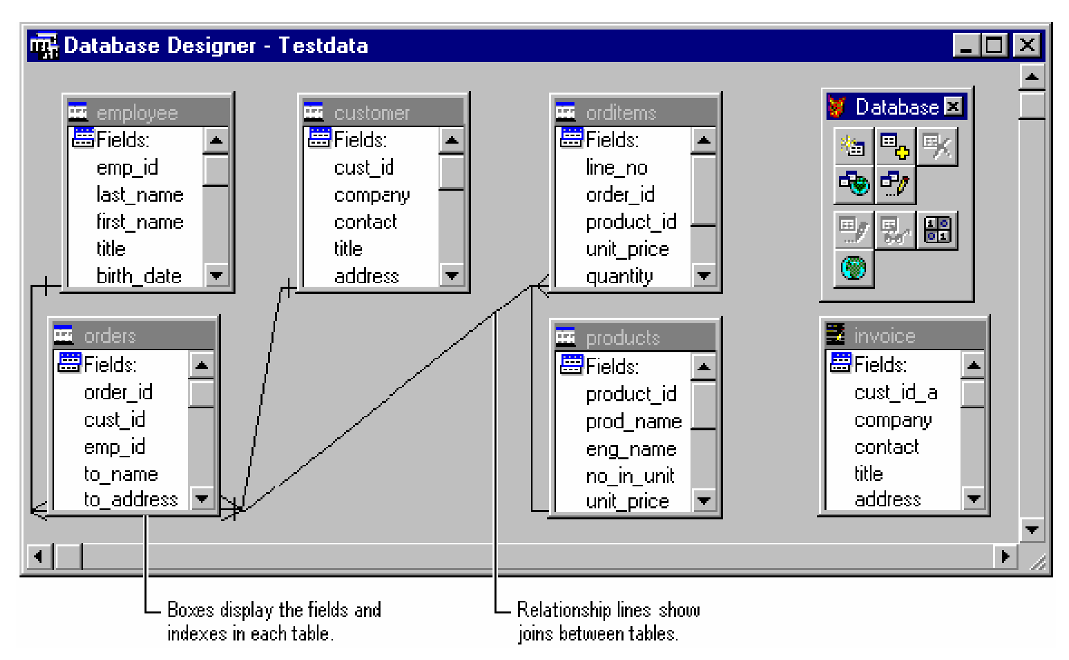
يتم الدخول في محتويات قاعدة البيانات لغرض التحديث عن طريق الإيعاز:-

modify database [database name]

فتظهر شاشة database designer

والتي يمكن من خلالها إنشاء الجداول والعلاقات والإجراءات.

ويبين الشكل (٣-١) مثال على قاعدة بيانات موجودة فيها عدد من الجداول المرتبطة بعلاقـــات مـــع بعـــضها البعض.

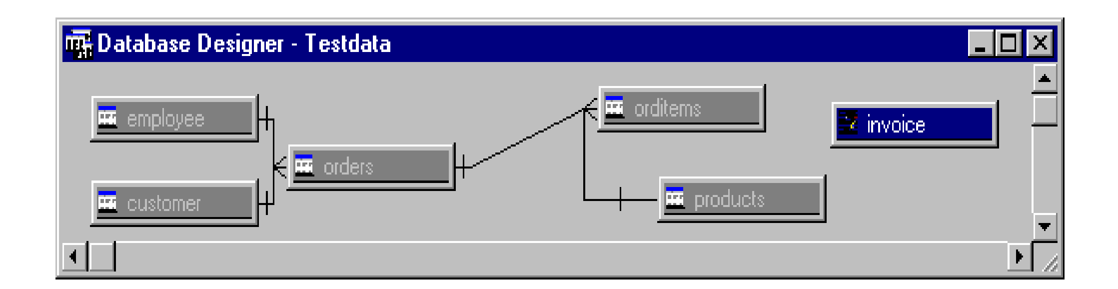



**الشكل (٣-١) مثال لقاعدة بيانات موجودة**

يمكن من خلال استخدام Toolbar Database من معالجة جداول قاعدة البيانات والعلاقات بين الجداول، كما ويمكن إظهار shortcut menus عن طريق نقر الزر الأيمن للفأرة في مساحة Database Designer.

يمكن تغيير حجم الجداول في القاعدة لعرض حقول الجدول والفهارس على الحقول كما ويمكن توسيع عرض الجدول أو كبس الجدول أيضا. ولغرض توسيع عرض الجدول نضع مؤشر الفأرة على الجدول المطلوب ثم نضغط الزر الأيمن فتظهر قائمة نختار منها Expand. أما إذا أردنا كبس عرض الجدول فإننا نختار Collapse.

أما إذا أردنا أن نوسع عرض كل الجداول أو نكبس عرض كل الجداول فإننا نضغط الزر الأيمن للفأرة في مساحة database designer ونختار Expand all أو Collapse all.




Collapse all tables in the database

من الممكن أن نغير توزيع الجداول في Database Designer أو أن نرتبها وفق صيغة معينة وذلك عن طريق اختيار Arrange من قائمة Database فتظهر شاشة لغرض اختيار نوع الترتيب المطلوب وحسب الجدول التالي:-

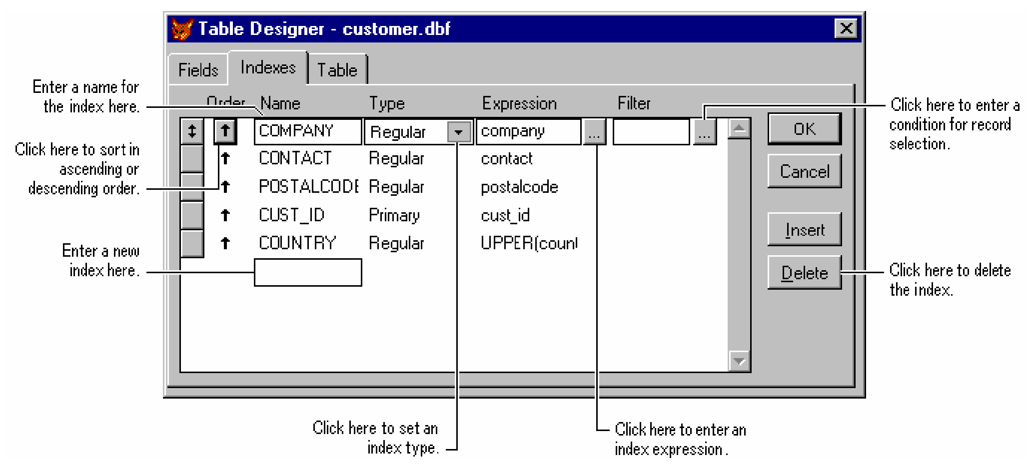
| نوع الترتيب | الاختيار المطلوب |
|---|---|
| Arrange the tables alphabetically by name | **By name** |
| Arrange the tables by type | **By type** |
| Align the tables in a row | **Horizontally** |
| Align the tables in a column | **Vertically** |
| Return the tables to their original size | **Resize objects to default height and width** |

لغرض خزن الملاحظات حول قاعدة البيانات نختار من قائمة database اختيار properties ونكتب الملاحظات في مربع comment.

## ج . فهرسة الجداول Table Indexing

الغرض من فهرسة الجداول هو التعامل السريع مع البيانات المخزونة فيه وترتيب عرض هذه البيانات في التقارير والمخرجات، وللفهرسة أهمية أكبر في بناء العلاقات بين الجداول.

يمكن إنشاء الفهارس على الحقول أو على تعبيرات، ولغرض إظهار أهمية الفهرسة يجب فهرسة الحقول التي يحصل عليها تصفية (filter) أو استفسارات أو تستخدم في إنشاء المنظورات Views أو في التقارير ويجب فهرسة الحقول التي تستخدم في إنشاء العلاقات بين الجداول. ولغرض إنشاء الفهارس نختار الجدول المطلوب فهرسته وندخل إلى شاشة Table Designer ونختار Indexes tab. كما في الشكل (٣-٢).




**الشكل (٣-٢) مثال لفهرسة جدول**

في مربع Name نطبع أسم الفهرس.

في قائمة Type نختار نوع الفهرس المطلوب.

في مربع Expression نطبع أسم الحقل المراد عمل فهرسة عليه أوبناء تعبير عند طريق ضغط مربع المحادثة dialog box الموجود في النهاية لإظهار Expression Builder.

في حالة اختيار مجموعة من القيود (عمل filter) ندخل التعبير في مربع Filter.

أخيرا نختار OK. لبناء الفهرسة على الجدول.

## د . اختيار نوع الفهرسة

من الممكن اختيار واحدة من أربعة أنواع من الفهارس (تظهر في قائمة Type) وهذه الأنواع الأربعة هي كالآتي:-

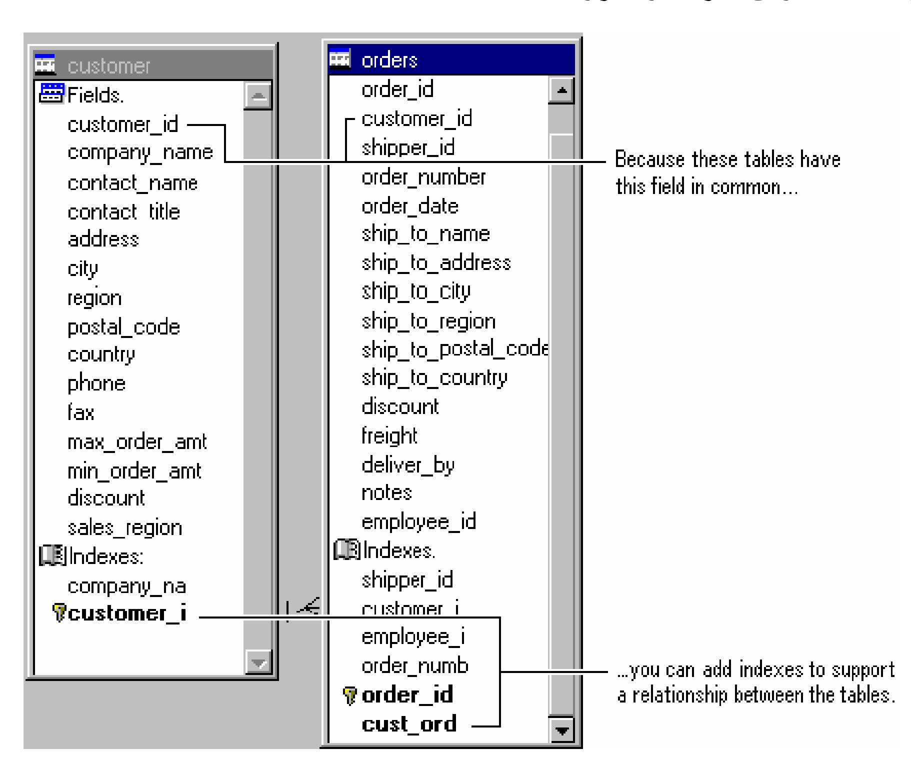
Primary
Candidate
Regular
Unique

**اولا . Primary Indexes:-** يضمن هذا النوع من الفهرسة وحدانية القيمة المدخلة في هذا الحقل (أو مجموعة من الحقول)، وبعبارة أخرى فإن الحقل المفهرس Primary هو الحقل المفتاحي للجدول، ويمكن عمل فهرسة واحدة فقط من هذا النوع لكل جدول ضمن قاعدة البيانات، وتظهر فهرسة الحقل على شكل Bold Font في الجدول للدلالة على Primary في شاشة Database Designer.

**ثانيا . Candidate Indexes:-** هذا النوع من الفهرسة يعطي أيضا إمكانية وحدانية القيمة المدخلة للحقل (أو الحقول) مثل Primary، ويمكن عمل أكثر من Index من هذا النوع للجدول الواحد وتكون أيضا على شكل Bold Font. وبعبارة أخرى فإن Candidate يعني أن هذا الحقل هو من الحقول المرشحة لأن تكون مفتاحية.

**ثالثا . Regular Indexes:-** يحدد الترتيب الذي تظهر فيه القيود تبعا لقيمة حقل الفهرسة عند معالجة الجدول سواء في الإدخال أو الإخراج أو التعامل، ويمكن إدخال قيم مكررة في حقل (أو حقول) الفهرسة، ويمكن عمل عدة أنواع من الفهارس من هذا النوع.

**رابعا . Unique Indexes:-** وضع هذا النوع من الفهرسة لعمل توافق (compatibility) مع النسخ السابقة، ويستخدم هذا النوع لعمل فهرسة على حقول فيها تكرار لإظهار القيمة الأولى لكل تكرار قيم للبيانات فقط وفق شرط أو شروط محددة.




**الشكل (٣-٣) مثال على علاقة بين جدولين**

**هـ. ربط الجداول مع بعضها البعض**

يوفر Database Designer عملية ربط الجداول مع بعضها العض بصورة سهلة وذلك عن طريق ربط فهارس الجداول مع بعضها. هذه العلاقات (relationships) التي يتم إنشاؤها في قاعدة البيانات يطلق عليها أسم العلاقات الثابتة (persistent relationships) وذلك لأنها تخزن كجزء من قاعدة البيانات، وفي كل مرة يتم استخدام الجداول في النماذج والاستفسارات والمنظورات تظهر هذه العلاقات بصورة بديهية بين الجداول دون الحاجة إلى تكرارها عن طريق العبارات البرمجية.

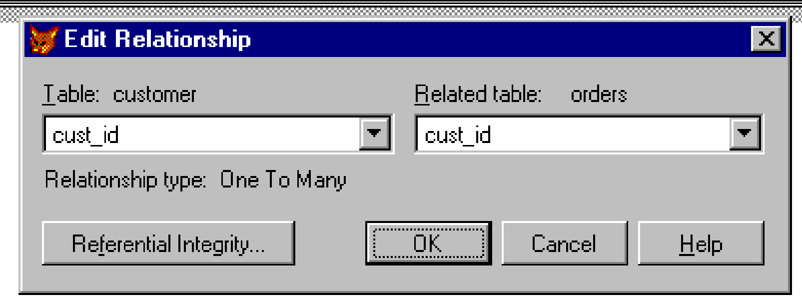
قبل أن ننشأ علاقة بين الجداول يجب أن تشترك الجداول مع بعضها في حقول وفهارس محددة. هذه الحقول تسمى Primary Key و Foreign Key. حقل Primary Key يعرف قيد واحد محدد في جدول معين، بينما يعرف Foreign Key القيود الموجودة في جدول آخر والتي تتساوى في قيمها مع Primary Key. لذلك نحتاج عند عمل Relation بين جدولين أن نفهرس Primary Key بصيغة Primary Index ونفهرس Foreign Key بصيغة Regular Index.

ويوضح الشكل (٣-٣) مثال على علاقة بين جدولين وهما customer و orders، كل منهما يحتوي على عدد من الحقول حسب تصميم القاعدة، ويوضح الشكل الحقول في الجدولين التي لها نفس القيمة (علاقة ١ : m محددة في تصميم القاعدة في نموذج البيانات) هذان الحقلان هما customer_id في جدول customer و customer_id في جدول orders (يمكن أن يحمل الحقلان أسمين مختلفين إلا أنه من الشرط أن يكونان بنفس نوع البيانات ونفس الحجم). ولغرض إنشاء علاقة بين الجدولين يجب أن تكون هناك فهرسة من نوع Primary على حقل customer_id في جدول customer على اعتبار أنه حقل مفتاحي في هذا الجدول (Primary Key)، ويجب أن تكون هناك فهرسة على حقل customer_id في جدول orders من نوع Regular على اعتبار أنه حقل أجنبي (Foreign Key) في هذا الجدول.

ويمكن بناء العلاقة بين الجدولين عن طريق تأشير Primary Key Inex في الجدول الأول وتحريك الفأرة دون رفع الضغط عن زر الفأرة (من الممكن عمل هذه العملية بضغط الزر الأيمن أو الأيسر للفأرة) ومن ثم تأشير Foreign Key Index في الجدول الثاني ومن ثم رفع الضغط عن زر الفأرة، فيتولد خط علاقة بين الجدولين يكون ذو طرف واحد من ناحية Primary Key Index وله ثلاثة تشعبات (للدلالة على Many) في ناحية Foreign Key Index وبهذه الطريقة يتم إنشاء علاقة الواحد للتعدد (one to many).

**و. تحديث العلاقة**

بعد إنشاء العلاقة بين جدولين يمكن الدخول في تحديث العلاقة وبناء قواعد وجوب الوحدانية ( Referential Integrity)، يتم ذلك عن طريق نقر خط العلاقة نقرتين مزدوجتين (double click) بواسطة زر الفأرة الأيسر لتظهر الشاشة التالية:-




تظهر الشاشة أعلاه طبيعة العلاقة بين الجدولين (وهي تكـون one-to-one إذا كانـت فهرسـة Primary و Foreign هما من نوع (Primary أو Candidate) وتكون العلاقـة (one-to-many) إذا كانـت فهرسـة Primary هي من نوع (Primary أو Candidate) وفهرسة Foreign هي من نوع (Regular).

ويمكن أيضا من خلال هذه الشاشة الدخول إلى وجوب التكاملية Referential Integrity والتي مـن خلالهـا يتم اختيار قواعد الإدخال والتحديث والحذف تبعا للجدول الأب وتأثيره على الجدول الابن.

عند الدخول في Referential Integrity تظهـر شاشـة فيها ثلاث tabs وهـي (rules for updating، rules for deleting، و rules for inserting) كل واحدة منها تضع قواعد خاصة بعملها وكالآتي:-

هذه القواعد هي :-

* Cascade:- أي تغيير في قيمة المفتاح الأساسي (Primary Key) في الجدول الأب سوف يـنعكس علـى القيم المقابلة لها في الجدول الابن، أي أنه إذا تغيرت قيمة المفتاح الأساسي في الجدول الأب فإن VFP سـوف يقوم بصورة أوتوماتيكية بتغيير القيم المقابلة في الجدول الابن إلى القيمة المحدثة.

* Restrict:- يمنع تحديث قيمة المفتاح الأساسي في الجدول الأب لذلك لا توجد قيود يتيمة في الجدول الابن.

* Ignore:- وهي القيمة Default وفيها يسمح بتغيير قيمة القيد في الجدول الأب حتى لو كان هنـاك قيـود مرتبطة معها في الجدول الابن (دون تغيير قيم الجدول الابن).

## <u>ز . فتح أكثر من قاعدة بيانات</u>

قد نحتاج في بعض التطبيقات إلى فتح أكثر من قاعدة بيانات، وذلك عـن طريـق تكـرار الإيعـاز Open Database، وفي هذه الحالة فإن آخر قاعدة تم فتحها تكون هي القاعدة الفعالة، ولغرض تحويل الفعالـة إلـى قاعدة أخرى مفتوحة (ولكنها غير فعالة) نستخدم إيعاز:-

Set Database to (database name )

***<u>تنبيه:-</u>*** *في VFP قد نفتح أكثر من قاعدة بيانات وخاصة في الاستفسارات والنماذج، لذا يجـب أولا اسـتخدام إيعاز Set Database To (...) قبل كتابة أي إيعاز آخر وذلك لضمان أننا نعمل مع القاعدة المطلوبة.*

**ح . غلق قاعدة البيانات**

يمكن غلق قاعدة بيانات مفتوحة من خلال إيعاز:-

```
Close database
```

والتي يجب أن يسبقها إيعاز Set Database To (...) إذا كان لدينا أكثر من قاعدة بيانات مفتوحة لضمان أننا نغلق قاعدة البيانات المطلوب غلقها فعلا. ويوضح المثال أدناه هذه العملية لقاعدة بيانات تحت أسم Testdata.

```
Set data to testdata
Close database
```

## ٤. إنشاء النماذج Creating Forms

تستخدم النماذج لإنشاء واجهات المستخدم التي تساعد العاملين على الأنظمة من إظهار البيانات وإدخالها إلى القاعدة بصورة سهلة وبكفاءة عالية. والحقيقة أن النماذج لها أفق أوسع من كونها واجهة عرض للمستخدم فقط وإنما يتعدى دورها إلى أنها عبارة عن حاوية تتضمن مجموعة من كائنات السيطرة (Control Objects) والتي تستجيب وفق صيغ محددة من قبل مصمم النظام إلى الأحداث التي تصدر من النظام أو من مستخدم النظام والتي تساعد في إنجاز إدارة بيانات النظام بكفاءة وبسهولة.

حيث يوفر VFP مصمم نماذج ذو كفاءة عالية يسهل ويسرع عملية إنشاء النماذج، حيث يوفر لمصمم النموذج الإمكانيات التالية:-

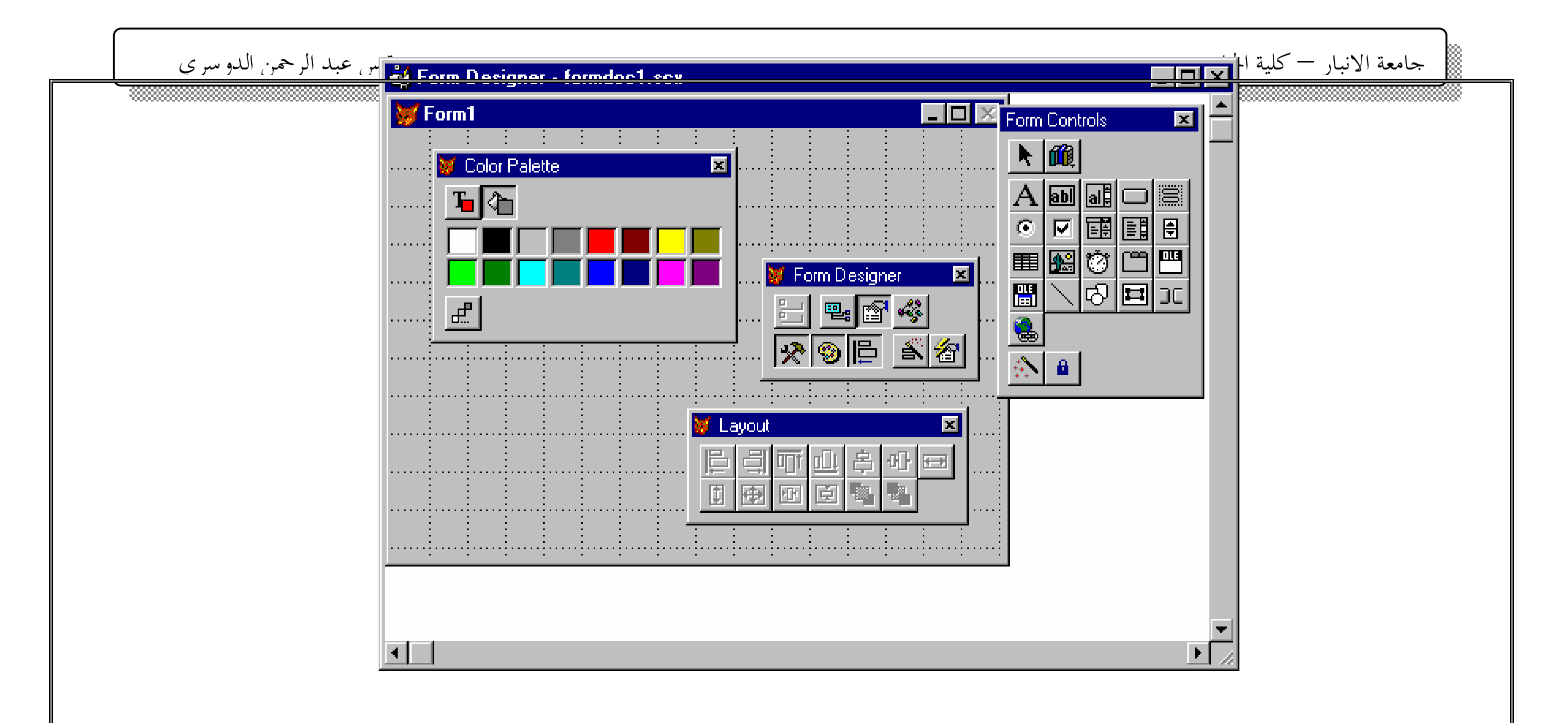
- عدد كبير ومختلف من الكائنات في النموذج.
- ربط البيانات في الجداول مع الكائنات.
- النموذج الرئيسي والنماذج التابعة.
- إمكانية معالجة عدد من النماذج مع بعضها البعض.
- إمكانية استخدام قوالب تصميم من VFP أو قوالب تصميم خاصة بالمصمم.

النماذج (Forms) ومجموعة النماذج (Form Set) هي عبارة عن كائنات (Objects) لها خاصيتها (Properties) وأحداثها (Events) وأسلوب التعامل مع الحدث (Methods) المستقلة والتي يمكن برمجتها وإعادة صياغتها وفقا للمصمم.

### أ. إنشاء نماذج جديدة Creating New Forms

يمكن إنشاء نماذج جديدة بعدة طرق:- إما عن طريق مصمم النماذج (Form Designer) أو قوالب التصميم (Form Wizards) أو عن طريق البرامج.

يمكن إنشاء النماذج باستخدام Form Designer عن طريق اختيار New من قائمة File ومن ثم اختيار Form وبعدها New File.



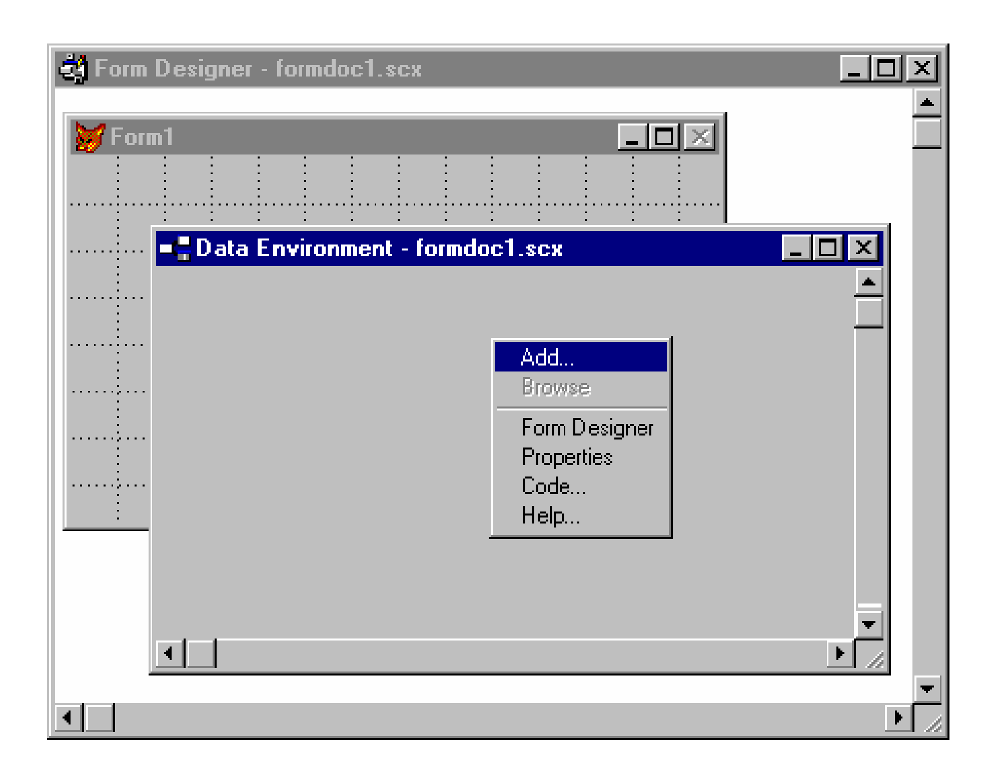

**الشكل (٤ - ١) مثال على نموذج فارغ مع أشرطة الأدوات (Toolbars) التي يمكن استخدامها ( Form Control، Form designer، Color، Layout)**

النموذج هو عبارة عن حاوية يمكن إضافة Objects من Classes مختلفة فيه (عن طريق استخدام Form Control Toolbar)، ويمكن تغيير خصائص هذه Objects عن طريق شاشة الخصائص ( Properties Window)، ويمكن ربط النموذج مع جداول البيانات من خلال بيئة البيانات (Data Environment).

## ب . التعامل مع بيئة بيانات النموذج Setting The Data Environment

كل Form أو Form Set يحتوي على بيئة بيانات خاصة به. وبيئة البيانات للنموذج هو عبارة عن كائن يحتوي على الجداول والمنظورات التي يتعامل معها Form. ويكن تصميم Data Environment بصورة مرئية Visually باستخدام مصمم بيئة البيانات Data Environment Designer وخزنه مع النموذج. ويقوم Data Environment بعملية فتح وغلق جداول البيانات بصورة أوتوماتيكية بالإضافة إلى المساعدة في تحديد مصدر السيطرة (Control Source) لكل كائن سيطرة (Control Object) في النموذج من خلال التعامل مع خاصية Control Source الموجودة في نافذة الخصائص ( Properties Windows).

ولغرض التعامل مع Data Environment التابع إلى Form مفتوح نختار من قائمة View اختيار Data Environment أو عن طريق نقر الزر الأيمن للفأرة لإظهار قائمة مختصرة نختار منها اختيار Data Environment. إذا تم اختيار قائمة View فسوف تظهر لنا شاشة محادثة باسم Open نختار منها الجدول أو المنظور لإضافته إلى Data Environment أو يمكن فتح هذه الشاشة عن طريق نقر الزر الأيمن لإظهار قائمة نختار منها Add.




**الشكل (٤ - ٢) مثال على Data Environment لنموذج مفتوح**

الجدول أدناه يبين عدد من الخصائص المشتركة لبيئة البيانات والتي يمكن التعامل معهـا مـن خـلال نافـذة الخصائص.

| الخاصيـــة **Property** | الوصـــــف **description** | القيمة البديهية**default Value** |
|---|---|---|
| AutoCloseTables | Controls whether tables and views are closed when the form is released. | True (.T(. |
| AutoOpenTables | Controls whether tables and views in the data environment are opened when the form is run. | True (.T(. |
| InitialSelectedAlias | Specifies the table or view that is selected when the form is run. | " "at design time. If not specified, at run time the first cursor added to the DataEnvironment is initially selected. |

## ج . إضافة جدول أو منظور إلى Data Environment

عند إضافة Table أو View إلى Data Environment يمكن أيضا اختيار الحقول والفهارس التابعـة إلـى Table أو View.

ولغرض إضافة Table أو View إلى Data Environment نتبع الخطوات التالية:-

من Data Environment designer نختار Add من قائمة الاختيارات.

من شاشة Add Table Or View نختار الجدول أو المنظور المطلوب من القائمة التي تظهر في الشاشة.

في حالة عدم فتح أي قاعدة بيانات أو في حالة الحاجة إلى استدعاء بيانات من خارج القاعدة نختار Other لاختيار مصادر البيانات من مكان آخر.

في حالة كون Data Environment فعالا تعرض شاشة الخصائص (Properties Window) الكائنات (Objects) والخصائص (Properties) المقترنة ببيئة البيانات. كل الجداول والمنظورات والعلاقات بل وحتى بيئة البيانات نفسها هي عبارة عن كائن مستقل يظهر في مربع Objects في Properties Window.

**د . حذف جدول من بيئة البيانات**

يمكن حذف Table من Data Environment، وكل العلاقات (relationship) التي تربط هذا الجدول سوف تحذف معه. ولغرض حذف الجدول من بيئة البيانات نتبع الخطوات التالية: -

في Data Environment Designer نختار الجدول المطلوب.

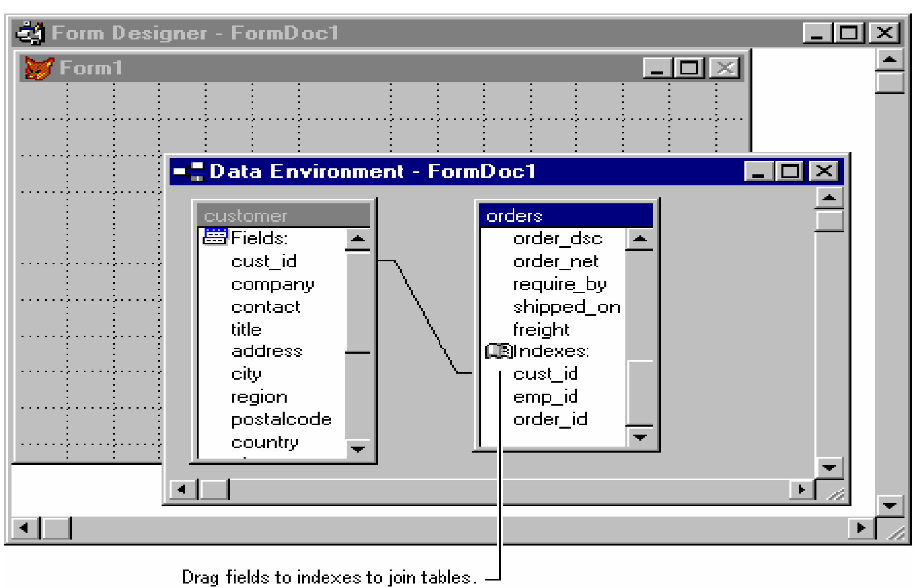
من قائمة Data Environment نختار Remove.

ويمكن أيضا عن طريق نقر الزر الأيمن للفأرة على الجدول المطلوب واختيار Remove من القائمة المختصرة التي ستظهر.

**هـ .تحديد العلاقات في Data Environment Designer**

عند إضافة جداول إلى Data Environment لها علاقات ثابتة في قاعدة البيانات، فإن هذه العلاقات ستضاف بصورة أوتوماتيكية إلى Data Environment. إما إذا كان الجدول لا يمتلك مثل هذه العلاقات الثابتة فيمكن إنشاء العلاقة في Data Environment عن طريق: -

نؤشر الحقل من الجدول الرئيسي مع ضغط زر الفأرة ونعمل له Drug (استمرار ضغط زر الفأرة) ونسقطه على الفهرسة المطلوبة في الجدول التابع كما في الشكل (٤ - ٣).




**الشكل (٤ - ٣) مثال لإنشاء علاقة بين جدولين في Data Environment**

**و . تحديث العلاقات في Data Environment Designer**

لغرض تحديث العلاقة نتبع ما يلي:-

في Properties Window نختار العلاقة (Relation) من خلال مربع Object. الخصائص المرتبطة بالعلاقة تقابل الإيعازات Set Relation و Set Skip.

خاصية RelationalExpr توضع فيه (Default) أسم الحقل المفتاحي (Primpary Key) للجدول الأب. وإذا كان الجدول التابع مفهرس على شكل تعبير (Expression) فإننا نحتاج إلى تغيير RelationalExpr إلى هذا التعبير.

إذا كانت العلاقة ليست من نوع One-To-Many نغير خاصية OneToMany إلى قيمة .F.، وهذا يقابل إيعاز Set Relation بدون Set Skip.

تغيير خاصية OneToMany إلى قيمة .T. يقابل استخدام إيعاز Set Skip، والذي يعني أنه عند الانتقال عبر قيود الجدول الأب (Table Parent) فإن مؤشر القيود سوف يبقى مؤشرا على نفس القيد في الجدول الأب إلى أن يتم انتقال المؤشر على جميع القيود المرتبطة في الجدول الابن (Child Table).

**ملاحظة:-** إذا كان لدينا علاقة One-To-Many فيجب أن تكون قيمة خاصية OneToMany هي .T. حتى لو كان لدينا علاقة ثابتة وموجودة (Persistent Relationship) في قاعدة البيانات.

**ز . إنشاء Single and Multiple Document Interface**

يوفر VFP إمكانية إنشاء نوعين من التطبيقات وهما:-

**Multiple Document Interface (MDI):-** تطبيق يحتوي على شاشة رئيسية واحدة وبقية الشاشات محتواة فيها أو عائمة على سطحها. وكمثال على MDI هو الشاشة الرئيسية لتطبيق VFP فهي تحتوي على Command Window، Edit Window، وعلى Designer Windows.

**Single Document Interface (SDI):-** تطبيق يحتوي على شاشة واحدة (أو عدد من الشاشات) المستقلة عن بعضها البعض في الظهور على شاشة سطح المكتب (Windows Desktop).

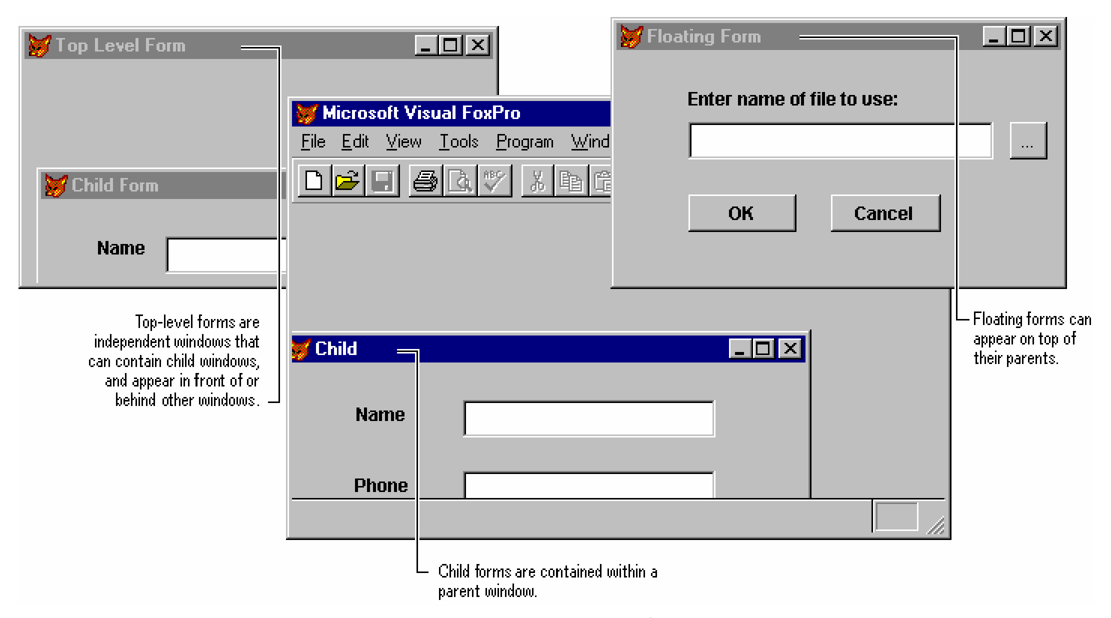
ولغرض إنشاء هذه الأنواع من الشاشات فإن VFP يوفر عدة أنواع من النماذج:-

**Child Form:-** هو Form محتوى ضمن شاشة أخرى، تستخدم في إنشاء MDI. وهذا النموذج لا يمكن تحريكه خارج حدود النموذج الأب (Parent Form)، وإذا تم تصغير نافذة الأب (minimize) فإن هذا Form يحصل له minimize أيضا.

**Floating Form:-** هو Form تابع إلى Parent Form ولكنه غير محتوى ضمنه، ويمكن تحريكه إلى أي موضع في الشاشة، وإذا تم تصغير نافذة الأب فإنه يصغر معه. يستخدم هذا النوع لإنشاء MDI.

**Top Level Form:-** هو Form مستقل بدون Parent Form يستخدم لإنشاء تطبيقات SDI أو لأن يكون Parent Form يحتوي على Forms أخرى.

ويوضح الشكل (٤-٤) أنواع Forms في VFP.




**الشكل (٤-٤) أمثلة على Forms**

## ح . تحديد نوع النموذج

يتم إنشاء جميع Forms بنفس الطريقة، ولكن يمكن تحديد نوعه من خلال خصائص معينة في نافذة الخصائص (Properties Window).

لتحديد Child Form نتبع ما يلي:-

* نغير قيمة خاصية ShowWindow إلى أحد القيم التالية:-

In Screen :- النموذج الأب هو النافذة الرئيسية للتطبيق VFP.

In Top-Level Form :- النموذج الأب هو النموذج الفعال عندما يتم استعراض النموذج الابن. يتم استخدام هذا الاختيار إذا أردنا أن يظهر النموذج الابن ضمن أي Top-Level Form.

* نغير خاصية MDIForm إلى .T. إذا أردنا أن يكون الابن مرتبط بالأب عند عمل maximize أو نغيره إلى .F. إذا أردنا أن يستقل النموذج الابن إلى نافذة منفصلة عند عمل maximize له.

لتحديد Floating Form نتبع ما يلي:-

* نغير قيمة خاصية ShowWindow إلى أحد القيم التالية:-

In Screen :-النموذج الأب هو النافذة الرئيسية للتطبيق VFP.

In Top-Level Form :- النموذج الأب هو النموذج الفعال عندما يتم استعراض النموذج الابن. يتم استخدام هذا الاختيار إذا أردنا أن يظهر النموذج الابن ضمن أي Top-Level Form.

* نغير خاصية Desktop إلى .T..

لتحديد Top-Level Form نتبع ما يلي:-

* نغير قيمة خاصية ShowWindow إلى ٢ - As Top-Level Form.

إخفاء الشاشة الرئيسية لتطبيق VFP

إذا كنا نعمل في Top-Level Form فإننا قد نرغب في عدم إظهار الشاشة الرئيسية للتطبيق VFP، ولغرض عمل هذا الشيء فإننا نتبع الخطوات التالية:-

* في الحدث (Event) المسمى Init داخل Form نكتب العبارة التالية:-

```
Application.Visible = .F.
```

* في الحدث المسمى Destory داخل Form نكتب العبارة التالية:-

```
Application.Visible = .T.
```

ويجب التأكد من وجود العبارة ThisForm.Release في أحد Events أو Methods التابعة إلى Form.

**ط . التوسع بالنماذج باستخدام Form Sets**

يمكن التعامل مع عدد من النماذج باستخدام form set والذي يمتلك الخصائص التالية:-

*من الممكن إظهار أو إخفاء كل Forms في نفس الوقت.

*من الممكن التعامل بصورة مرئية (Visually) مع عدد من Forms للسيطرة على مواقعها.

*لكون كل Forms في Form Set تشترك في ملف شاشة واحد (.scx) لذا كل Forms تشترك في Data environment واحدة. ولهذا السبب إذا حركنا record pointer في أحد Forms فإن هذا التحريك سينعكس في بقية Forms التي تنتمي إلى نفس Form Set.

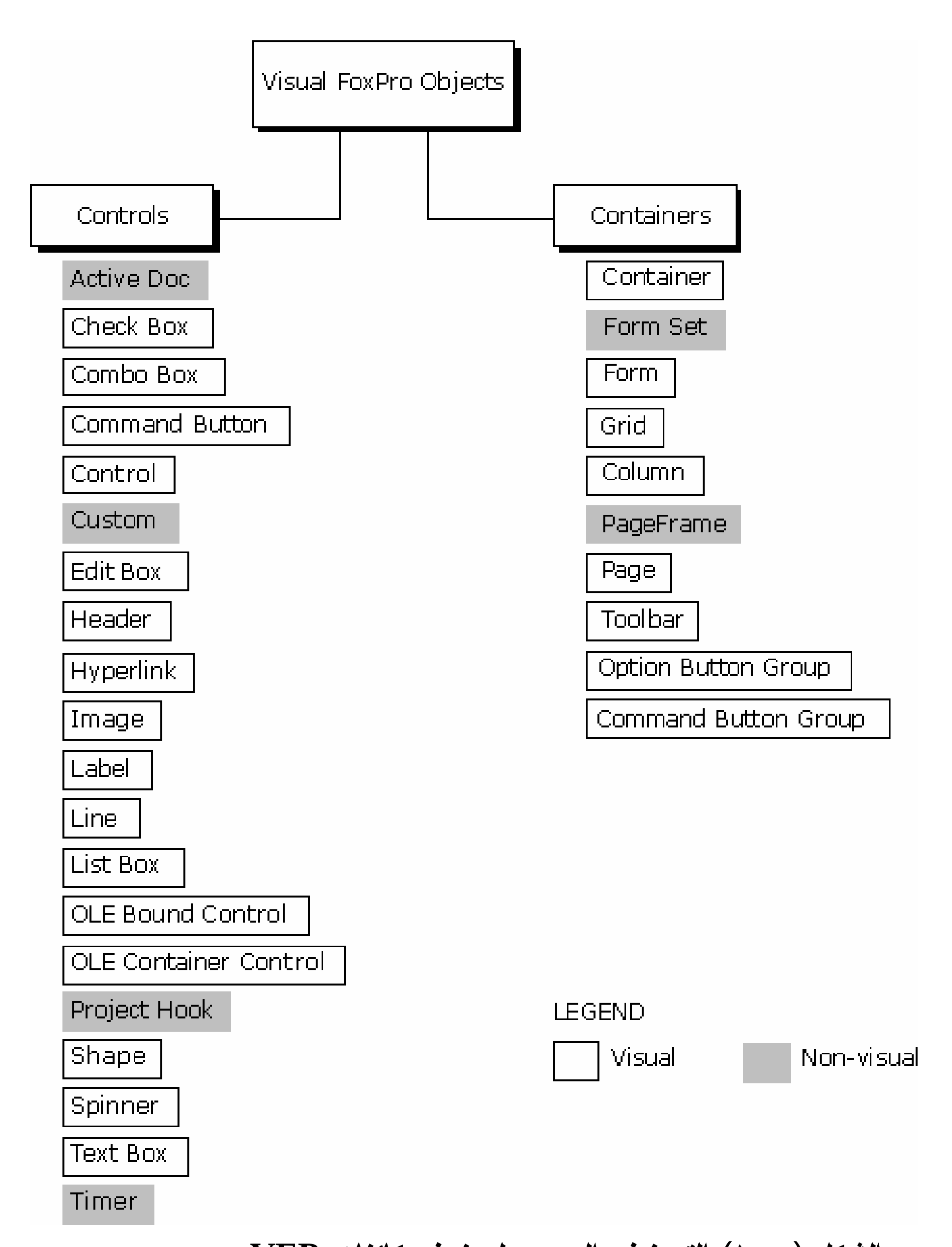
يمكن إنشاء Form Set عن طريق ما يلي:-

من قائمة اختيار Form نختار Create Formset.

ولغرض إضافة Forms إلى Form Set، من قائمة اختيار Form نختار Add New Form. ولحذف Form موجود داخل Form Set، من مربع Form الموجود أسفل Form Designer نختار Form، ثم من قائمة اختيار Form نختار Remove Form.

## ٥ . مسيطرات النماذج Form Controls

الشكل (٥-١) يبين التصنيف الهرمي لصنوف كائنات VFP.




**الشكل (٥-١) التصنيف الهرمي لصنوف كائنات VFP**

من الشكل (٥-١) يتبين لنا أن صنوف VFP تقسم إلى قسمين رئيسيين وهما Container Classes & non-Container Classes وبشكل اخر يمكن رؤية الصنوف من وجهة نظر أخرى حيث يمكن أن يحتـوي Form على نوعين من Controls :- النوع الأول من Controls يرتبط مع Data والنوع الثاني غير مرتبط مـع Data. فعند تعامل المستخدم مع Controls مرتبطة مع Data فقيمة هذا Control سوف تخزن في مصدر بيانات المسيطر (Data Source) والذي يمكن أن يكون حقل في جدول (Table Field) أو متغير (Variable). يتم ربط Control

مع Data عن طريق التعامل مع خاصية (ControlSource)، أو في حالة كون المسيطر من نوع Grid فالخاصية التي تربط المسيطر مع البيانات هي (RecordSource).

تتميز Controls في VFP بمرونتها عند الاستخدام. لذلك هناك عدد من المسيطرات التي يمكن استخدامها لإنجاز عمل محدد، ولهذا السبب فإنه يجب إيجاد أسلوب ثابت لاستخدام المسيطرات حتى يتمكن مستخدم النظام من معرفة وتوقع ما يمكن أن ينجزه عند رؤيته لواجهة الاستخدام والشاشات المصممة من قبل مصمم النظام. فمثلا LABEL يوجد فيه الحدث Click Event، ولكن المألوف لدى المستخدمين أنه يتوقع أن Click Event هو عن طريق COMMAND BUTTONS.

معظم العمليات التي يرغب مصمم النظام من درجها في النماذج تقع ضمن واحدة من الأقسام التالية:-

إعطاء المستخدم مجموعة من الخيارات المحددة مسبقا.

قبول إدخال المستخدم للبيانات التي لا يمكن تحديدها مسبقا.

قبول إدخال المستخدم للبيانات ضمن مديات محددة.

السماح للمستخدم من إجراء فعاليات محددة.

إجراء فعاليات محددة في أوقات معينة.

عرض المعلومات.

**أ . استخدام Option Button Groups**

هو عبارة عن Container يحتوي على عدد من الخيارات، يتيح للمستخدم تأشير أحد الخيارات بدلا من إدخال البيانات إلى القاعدة لضمان عدم حصول خطأ في الإدخال.

عند إنشاء Option Group فإن القيمة البديهية لعدد الخيارات فيه هو (٢)، ويمكن تغيير عدد الخيارات عن طريق تغيير قيمة الخاصية ButtonCount.

الخاصية Value يؤشر رقم الاختيار الذي تم تحديده من قبل المستخدم، فمثلا إذا اختار المستخدم الخيار الرابع فإن Value لهذا option group سوف يساوي (٤).

إذا كانت الخاصية ControlSource هو عبارة عن Character فإن قيمة الخاصية Value سوف تكون عنوان الاختيار (Caption) وليس رقمه.

يمكن التعامل مع كل خيار داخل option group على حدة، ويمكن تغيير خصائص كل خيار أو كتابة الإيعازات داخل الصيغ (Methods) التابعة لكل خيار على حدة.

يمكن تغيير خصائص الخيارات أثناء وقت التصميم (Design Time)، فمثلا إذا أدرنا أن نغير عنوان (Caption) الخيار المسمى optCust في option group اسمه opgChoice فنكتب الإيعاز التالي:-

```
ThisForm.opgChoice.optCust.Caption = "Sort by Customer"
```

كما ويمكن تغيير خصائص الخيارات أثناء وقت التنفيذ (Run Time) عن طريق استخدام الخاصية Buttons عن طريق تحديد رقم الخيار، فمثلا إذا كان optCust هو ثالث خيار في option group المسمى opgChoice فنكتب الإيعاز التالي:-

ThisForm.opgChoice.Buttons(3).Caption = "Sort by Customer"

ولتغيير خاصية محددة لكل الخيارات في option group نستخدم الصيغة (method) المسمى SetAll، فمثلا إذا أردنا أننعل Disable لكل الخيارات في option group اسمه opgMyGroup فنكتب الإيعاز التالي:-

ThisForm.opgMyGroup.SetAll("Enabled",.F.,"OptionButtons")

عندما تكون الخيارات (Buttons) في حالة Disable فإنها تظهر بلون خلفية يمكن تغييره من خلال الخاصية DisabledBackColor وبلون أمامي يمكن تغيير من خلال الخاصية DisabledForColor. كما ويمكن عمل الخاصية Enabled التابعة إلى option group للقيمة .F. وينتج عنه أن كل الخيارات تكون غير فعالة (disabled).

يمكن استخدام الخاصية Value لتحديد أي خيار تم تأشيره، فإذا كانت الخاصية Control Source من نوع بيانات Numeric، وكان لدينا option group مكون من خمسة خيارات وتم ضغط الخيار الثالث مثلا فإن Value ستكون مساوي إلى (٣) وإذا لم يتم ضغط أي خيار فإن Value ستكون مساوية إلى (صفر).

## ب . استخدام List Box و Drop Down List Box

المسيطرات List Box و Drop-Down List Box (هو Combo Box فيه خاصية Style مساوية إلى ٢ - DropdownList)، تتيح للمستخدم قائمة متحركة scrollable list تحتوي على عدد من الخيارات أو أجزاء من المعلومات. الاختلاف بينهما أنه في list box يوجد عدد محدد من العناصر (items) ممكن ان تكون مرئية دائما في كل وقت، بينما في drop down list box هناك فقط عنصر واحد يكون مرئيا، إلا أن المستخدم يستطيع أن يعمل click لكي يستعرض قائمة متحركة لعرض عناصر drop down list box.

ويوضح الجدول أدناه بعض خصائص list box و drop down list box والتي غالبا ما يتم تغيرها أثناء Design Time.

| Property | Description |
|---|---|
| **ColumnCount** | The number of columns in the list box. |
| **ControlSource** | Where the value that a user chooses from the list is stored. |
| **MoverBars** | Whether mover bars are displayed to the left of list items so that a user can easily rearrange the order of items in the list. |
| **Multiselect** | Whether the user can select more than one item in the list at a time. |
| **RowSource** | Where the values displayed in the list come from. |
| **RowSourceType** | Whether the RowSource is a value, a table, a SQL statement, a query, an array, a list of files, or a list of fields. |

جدول يوضح بعض خصائص list box / drop down list box

**ملاحظة:-** الخاصية Value التابعة إلى List يمكن أن تكون numeric أو character. والقيمة البديهية هي numeric. وإذا كانت الخاصية RowSource من نوع character فنضع قيمة فارغة (empty) في الخاصية Value إذا كنا نريد أن تعكس الخاصية Value بيانات الخيار المنتخب. ويمكن إدخال قيمة فارغة عن طريق ضغط SPACEBAR وبعدها BACKSPACE لهذه الخاصية في Properties window.

يوضح الجدول أدناه بعض الصيغ (methods) بالنسبة إلى list box.

| Method | Description |
|---|---|
| **AddItem** | Adds an item to a list with a RowSourceType of 0. |
| **RemoveItem** | Removes an item from a list with a RowSourceType of 0. |
| **Requery** | Updates the list if the values in the RowSource have changed. |

يمكن إملاء list box بعناصر من مصادر بيانات مختلفة عن طريق تغيير الخاصية RowSourceType والخاصية RowSource.

الخاصية RowSourceType تحدد نوع مصدر البيانات الذي يغذي list box (مثلا table أو array ...الخ).
وعند تحديد نوع المصدر يمكن بعدها تحديد المصدر نفسه عن طريق الخاصية RowSource.

ويوضح الجدول أدناه أنواع مصادر البيانات بالنسبة إلى List Box.

| RowSourceType | Source of the List Items |
|---|---|
| ٠ | None. Programmatically add items to the list. |
| ١ | Value |
| ٢ | Alias |
| ٣ | SQL Statement |
| ٤ | Query (.qpr( |
| ٥ | Array |
| ٦ | Fields |
| ٧ | Files |
| ٨ | Structure |
| ٩ | Popup. Included for backward compatibility. |

أدناه توضيح لأنواع مصادر البيانات بالنسبة إلى list box.

**None:-** إذا كان RowSourceType يساوي صفر، نستطيع إضافة عناصر إلى list عن طريق الصيغة AddItem وكما في المثال التالي الذي يضيف العناصر إلى list اسمه lstMyList داخل النموذج الذي اسمه frmForm1:-

```
frmForm1.lstMyList.RowSourceType = 0
frmForm1.lstMyList.AddItem("First Item")
frmForm1.lstMyList.AddItem("Second Item")
frmForm1.lstMyList.AddItem("Third Item")
```

والصيغة RemoveItem يتيح لنا حذف العناصر من list وكما في الإيعاز التالي:-

```
frmForm1.lstMyList.RemoveItem(2)
```

**Value**:- إذا عملنا RowSourceType مساوية إلى (١)، فإننا نستطيع وضع قيم متعددة للخاصية RowSource وكما في المثال التالي:-

```
Form1.lstMyList.RowSourceType = 1
Form1.lstMyList.RowSource = "one,two,three,four"
```

**Alias**:- إذا عملنا RowSourceType مساوية إلى (٢)، فإننا نستطيع إدراج حقل واحد أو أكثر من جدول مفتوح. فإذا كانت الخاصية ColumnCount مساوية إلى (٠ أو ١) فإن أول حقل في الجدول سيظهر في list box، وإذا تم تغييره مثلا إلى (٣) فإن أول ثلاثة حقول ستظهر في list box وهكذا.

**SQL Statement**:- إذا عملنا RowSourceType مساوية إلى (٣)، نستطيع وضع SELECT-SQL في خاصية RowSource، والمثال التالي يوضح هذا:-

```
SELECT * FROM Customer INTO CURSOR mylist
```

إذا تم تغيير RowSource عن طريق البرمجة، يجب تذكر أن نضع SELECT داخل quotation mark.

**Query**:- إذا عملنا RowSourceType مساوية إلى (٤)، فنستطيع أن نجعل مصدر بيانات list box هو عبارة عن نتيجة استفسار (Query) مصمم باستخدام Query Designer. والمثال التالي يوضح ذلك:-

```
THISFORM.List1.RowSource = "region.qpr"
```

**Array**:- إذا عملنا RowSourceType مساوية إلى (٥)، يمكن تغذية list box من مصفوفة (array) تم إنشاؤها من أي مكان آخر في التطبيق.

***ملاحظة : تجنب قدر الإمكان استخدام array كمصدر بيانات، لأنها تسبب الكثير من المشاكل).***

**__Fields:-__** إذا عملنا RowSourceType مساوية إلى (٦)، نستطيع تغذية list box عن طرق حقل من جدول أو عدة حقول مفصولة بفوارز.

عند استخدام fields نستطيع وضع المعلومات التالية في خاصية Rowsource:-

field
alias.field
alias.field,field,…

على العكس من (٢) يمكن درج حقول الجدول هنا بغض النظر عن ترتيبها الأصلي في الجدول.

**__Files:-__** إذا عملنا RowSourceType مساوية إلى (٧)، فسيتم تغذية list box من الملفات الموجودة في المكتبة الحالية (current directory)، إضافة إلى وجود خيارات أخرى لتغيير القرص ولتغيير المكتبة. كما في الشكل أدناه.




الشكل(٥-٣) list box قيمة الخاصية RowSourceType فيه تساوي ٧

إذا أردنا أن نستعرض ملفات ذات استطالة محددة، نغير خاصية RowSource إلى الاستطالة المطلوبة (مثلا DBF ).

**__Structure:-__** إذا عملنا RowSourceType مساوية إلى (٨)، فسيتم تغذية list box من حقول في جدول يتم تحديده عند تغيير RowSource. هذا النوع يكون مفيدا إذا أردنا أن يقوم المستخدم بالبحث عن حقل معين لترتيب الجدول عليه.

**__Popup:-__** هذا النوع وضع لأجل التوافق مع الإصدارات القديمة للتطبيق VFP.

**__إنشاء List Box ذات أكثر من عمود__**

القيمة البديهية لعدد الأعمدة في list box هو (١)، إلا أنه يمكن إنشاء list box ذات أكثر من عمود وذلك عن طريق تغيير خاصية ColumnCount إلى عدد الأعمدة المطلوب، وتغيير خاصية ColumnWidths إلى المساحة المطلوب إظها كل حقل فيه. كما في المثال:-

```
THISFORM.listbox.ColumnWidths = "10, 15, 30"
```

وبعدها نغير خاصية RowsourceType إلى (٦ - Fields). وأخيرا نغير الخاصية RowSource إلى الحقول المطلوب عرضها. كما في المثال:-

```
form.listbox.RowSource = "contact,city,country"
```

## ج . استخدام Check Box

يمكن استخدام Check box للسماح للمستخدم بالتعامل مع حالة Boolean (True & False).
هناك فقط أربعة حالات لقيمة خاصية Value بالنسبة إلى Check box، ويوضح الجدول أدناه هذه الحالات الأربعة.

حالات خاصية Value بالنسبة إلى Check box

| | |
|---|---|
| Check1 | 0 or .F. |
| Check2 | 1 or .T. |
| Check3 | 2 |
| Check4 | .NULL. |

**ملاحظة:-** يمكن للمستخدم إعطاء قيمة NULL إلى Check box عن طريق ضغط Ctrl+0.

إذا أعطينا الخاصية ControlSource بالنسبة إلى Check box بأن تشير إلى حقل منطقي في جدول بيانات (logical field)، فإن check box سوف يظهر على شكل مربع داخله علامة صح إذا كانت قيمة الحقل .T. ويظهر مربع فارغ إذا كانت قيمة الحقل .F. ويظهر مربع فارغ بلون رصاصي غامق إذا كانت القيمة .NULL

## د . استخدام Text Box

المسيطر text box هو المسيطر الأساسي الذي يسمح للمستخدم من إضافة وتحديث قيم البيانات لكل أنواع البيانات ما عدا Memo.

ولتأشير أو تغيير العبارة التي تظهر في text box نقوم أولا بتأشير خاصية Value. إذا قمنا بتغيير قيمة خاصية ControlSource بالنسبة إلى text box فإن القيمة المعروضة فيه سوف تخزن في الخاصية Value وكذلك في الحقل أو المتغير المقترن بالخاصية ControlSource.

ولغرض التحقق من القيمة المدخلة في text box نكتب عبارات السيطرة في الصيغة (method) المقترنة بالحدث Valid. فإذا كانت القيمة الداخلة خاطئة فإن قيمة الرجوع سيكون (.F.) أو ٠. وإذا رجعت قيمة (.F.) ستظهر عبارة "Invalid Input". وإذا أردنا أن نظهر العبارة الخاصة بنا نستخدم WAIT WINDOW أو MESSAGEBOX() في عبارات Valid ونرجع ٠. فمثلا إذا كان لدينا text box ندخل فيه قيمة تاريخ في المستقبل، فإننا نستطيع التحقق من ذلك عن طريق كتابة العبارات التالية في الحدث Valid الخاص بالمسيطر text box، وكما يلي:-

```
IF CTOD(THIS.Value) < DATE( )
   = MESSAGEBOX("You need to enter a future date",1)
   RETURN 0
ENDIF
```

وإذا أردنا أن نختار كل العبارة في text box عندما يحصل عليه تأشير نغير قيمة الخاصية SelectOnEntry.
كما ونستطيع استخدام خاصية InputMask لتحديد القيمة التي يمكن إدخالها في text box ونستخدم خاصية Format لتحديد كيفية ظهور العبارة في text box.
وإذا أردنا أن لا تظهر العبارة التي ندخلها إلى text box (مثل واجهات كلمات المرور Password) نغير قيمة خاصية PasswordChar بالنسبة إلى text box إلى * أو أي رمز حرفي عام آخر.
كما ويمتلك text box عدة خصائص يمكن تغييرها لتسهيل عملية إدخال قيم التاريخ والجدول أدناه يوضح هذه الخصائص.

| **Property** | **Description** |
|---|---|
| Century | Whether the first two digits of the year are displayed or not. |
| DateFormat | Format the date in the text box to one of fifteen predetermined formats, such as American, German, Japanese. |
| StrictDateEntry | Setting StrictDateEntry to 0 - Loose allows a user to enter dates in more flexible formats than the default 99/99/99. |

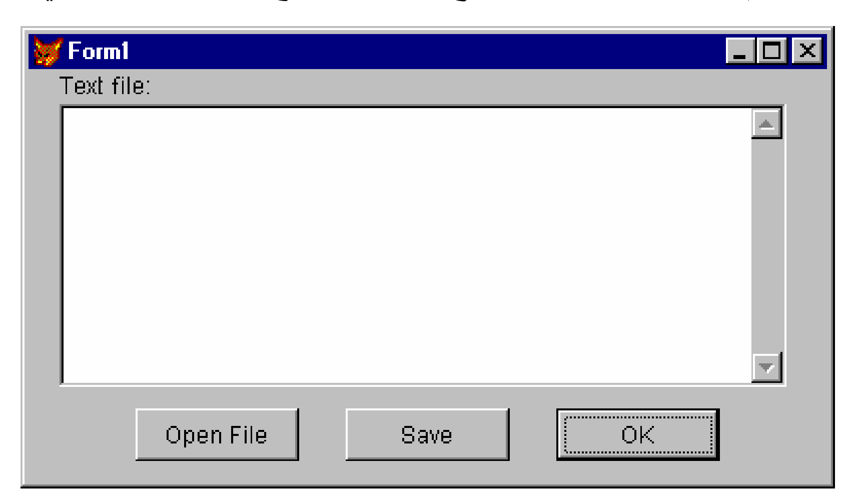
ويبين الجدول أدناه بعض الخصائص بالنسبة إلى text box والتي غالبا ما يتم التعامل معها.

| Property | Description |
|---|---|
| Alignment | Whether the contents of the text box are left justified, right justified, centered, or automatic. Automatic alignment depends on the data type. Numbers, for example, are right justified and characters are left justified. |
| ControlSource | The table field or variable whose value is displayed in the text box. |
| InputMask | Specifies the data entry rule each character entered must follow. For specific information about InputMask, see Help. |
| SelectOnEntry | Whether the contents of the text box are automatically selected when the text box receives the focus. |
| TabStop | Whether the user can tab to the control. If TabStop is set to .F., a user can still select the text box by clicking it. |

## هـ . استخدام Edit Box

عند استخدام edit box فإننا نسمح للمستخدم من التعامل مع حقول Memo أو الحقول من نوع character الطويلة جدا. Edit box له عدد من الميزات منها السماح بإدخال البيانات بحرية والتنقل عبرها باستخدام الأسهم أو page up و page down وباستخدام scrollbars.

نستطيع السماح للمستخدم من تحديث ملف من نوع text والنموذج أدناه يمثل هذا الشيء.




**الشكل (٥ - ٤) مثال حول استخدام Edit Box لتحديث ملف من نوع text**

الاختيار OK يستخدم لإغلاق النموذج ونكتب فيه العبارة

ThisForm.Release أو Release ThisForm

والاختياران الآخران File open و Save يستخدمان لفتح الملف والآخر لخزن الملف بعد تحديثه. (ليكن اسم الاختيار File open هو cmdOpenFile والاختيار Save هو cmdSave).

الجدول التالية يبين العبارات البرمجية التي تكتب في الحدث Click الخاص بالاختيار File open.

| Code | Comments |
|---|---|
| `CREATE CURSOR textfile ;`<br>`(filename c(35), mem m)`<br>`APPEND BLANK` | Create a cursor with a character field to hold the name of the text file and a memo field to hold the contents of the text file.<br><br>Add a blank record to the cursor. |
| `REPLACE textfile.FileName WITH ;`<br>`GETFILE("TXT")` | Use the GETFILE( ) function to return the name of the file to open. Store the name in the FileName field of the cursor. |
| `IF EMPTY(textfile.FileName)`<br>`RETURN`<br>`ENDIF` | If the user chooses Cancel in the Get File dialog box, the FileName field will be empty and there will be no file to open. |
| `APPEND MEMO mem FROM ;`<br>`(textfile.FileName) OVERWRITE` | Fill the memo field with the text in the file. |
| `THISFORM.edtText.ControlSource = ;`<br>`"textfile.mem"`<br>`THISFORM.Refresh` | Set the ControlSource of the edit box on the form. |
| `THISFORM.cmdSave.Enabled = .T.` | Enable the Save button. |

والجدول التالي يحتوي على العبارات التي تكتب في الحدث Click الخاص بالاختيار Save.

| Code | Comments |
|---|---|
| `COPY MEMO textfile.mem TO ;`<br>`(textfile.filename)` | Overwrites the old value in the file with the text in the memo field. |

والجدول التالي يبين بعض خصائص Edit Box والتي غالبا ما تستخدم عند التعامل معه.

| Property | Description |
|---|---|
| AllowTabs | Whether the user can insert tabs in the edit box instead of moving to the next control. If you allow tabs, be sure to indicate that users can move to the next control by pressing CTRL+TAB. |
| HideSelection | Whether selected text in the edit box is visibly selected when the edit box doesn't have the focus. |
| ReadOnly | Whether the user can change the text in the edit box. |
| ScrollBars | Whether there are vertical scrollbars. |

## و . استخدام Combo Box

combo box يمتلك خواص list box وخواص text box معا. هناك نوعان من combo box وهما Drop-down combo و Drop-down list. ويمكن تحديد نوع combo box عن طريق تغيير الخاصية Style.

يمكن إضافة عنصر جديد في drop-down combo عن طريق كتابة العبارة التالية في الصيغة (method) المقترنة بالحدث Valid وكما يلي:-

```
This.AddItem(This.Text)
```

على العموم قبل إدخال قيمة جديدة من الفضل التحقق من عدم وجود هذه القيمة مسبقا في عناصر combo box وكما يلي:-

```
lItemExists = .F. && assume the value isn't in the list.
FOR i = 1 to THIS.ListCount
  IF THIS.List(i) = THIS.Text
    lItemExists = .T.
    EXIT
  ENDIF
ENDFOR
IF !ItemExists
  THIS.AddItem(THIS.Text)
ENDIF
```

الجدول التالي يبين بعض خصائص combo box والتي يتم التعامل معها غالبا.

| **Property** | **Description** |
|---|---|
| ControlSource | Specifies the table field where the value that the user chooses or enters is stored. |
| DisplayCount | Specifies the maximum number of items displayed in the list. |
| InputMask | For drop-down combo boxes, specifies the type of values that can be typed in. |
| IncrementalSearch | Specifies whether the control tries to match an item in the list as the user types each letter. |
| RowSource | Specifies the source of the items in the combo box. |
| RowSourceType | Specifies the type of the source for the combo box. The RowSourceType values for a combo box are the same as for a List. For an explanation of each, see Help or the discussion on list boxes earlier in this chapter. |
| Style | Specifies whether the combo box is a drop-down combo or a drop-down list. |

**ز . استخدام Spinner**

يستخدم spinner للسماح للمستخدم اختيار قيمة معينة من بين مجموعة من القيم أو طبع القيمة مباشرة داخل spinner.

يتم تغيير قيمة الخاصية KeyboardHighValue والخاصية SpinnerHighValue وتوضع فيهما أعلى قيمة يمكن للمستخدم إدخاله في spinner.

وبنفس الحالة بالنسبة للخاصية KeyboardLowValue والخاصية SpinnerLowValue حيث توضع فيهما أقل قيمة يمكن للمستخدم إدخاله في spinner.

والجدول التالي يبين بعض خصائص spinner والتي غالبا ما يتم التعامل معها.

| Property | Description |
|---|---|
| Interval | How much to increment or decrement the value each time the user clicks the Up or Down buttons. |
| KeyboardHighValue | The highest value that can be entered into the spinner text box. |
| KeyboardLowValue | The lowest value that can be entered into the spinner text box. |
| SpinnerHighValue | The highest value that the spinner will display when the user clicks the Up button. |
| SpinnerLowValue | The lowest value that the spinner will display when the user clicks the Down button. |

## ح . استخدام Command Buttons و Command Buttons Group

من أكثر المسيطرات استخداما في التطبيقات، وغالبا ما توضع العبارات البرمجية فـي الحـدث Click داخـل Command buttons.

ويبين الجدول التالي بعض خواص Command Button الأكثر شيوعا.

| Property | Description |
|---|---|
| Cancel | Specifies that the code associated with the Click event of the command button executes when a user presses ESC. |
| Caption | Text displayed on the button. |
| DisabledPicture | The .bmp file displayed when the button is disabled. |
| DownPicture | The .bmp file displayed when the button is pressed. |
| Enabled | Whether the button can be chosen. |
| Picture | The .bmp file displayed on the button. |

ويمكن أيضا إدراج command buttons في group بحيث يمكن معالجتهم كلهم مع بعض أو كل واحد علـى حدة. وإذا أردنا أن نعمل مع صيغة (method) واحدة في الحدث Click التـابع إلـى Command buttons group، فإن الخاصية Value التابعة إلى group سوف يكون فيه رقم Command Button الذي حدث عليـه Click. والمثال التالي يوضح هذا:-

```
DO CASE
  CASE THIS.Value = 1
WAIT WINDOW "You clicked " + THIS.cmdCommand1.Caption; NOWAIT
    * do some action
  CASE THIS.Value = 2
WAIT WINDOW "You clicked " + THIS.cmdCommand2.Caption; NOWAIT
    * do some other action
  CASE THIS.Value = 3
WAIT WINDOW "You clicked " + THIS.cmdCommand3.Caption; NOWAIT
```

* do a third action
ENDCASE

**ملاحظة:-** إذا كتبنا عبارات برمجية داخل الحدث Click التابع لأحد Buttons ضمن command buttons group فإن هذه العبارات ستنفذ حتى ولو كان هناك عبارات برمجية داخل الحدث Click التابع إلى command buttons group.

## ط . استخدام Timer Control

يستخدم Timer Control لتنفيذ فعاليات معينة في أوقات محددة دون تدخل المستخدم.

كل Timer توجد فيه خاصية Interval والذي يحدد عدد milliseconds التي بين حدث وحدث آخر. خاصية Interval فيها بعض التحديدات عند برمجتها وهي:-

Interval هو بين ٠ إلى ٢١٤٧٤٨٣٦٤٧ والذي يعني أن أكبر Interval مسموح به هو ٥٩٦,٥ ساعة (أكثر من ٢٤ يوم).

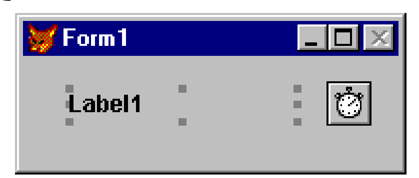
ليس مضمونا دائما أن يكون Interval متساويا، ولضمان الدقة يجب التحقق من clock system داخل الحاسبة.

إذا كان التطبيق يحتوي على تطبيقات أخرى تعمل حملا كبيرا على الحاسبة مثل long loop، معالجة Disk، معالجة Port، معالجة Network قد تؤدي إلى إيقاف Timer.

ويبين الجدول التالي أهم خواص Timer.

| Property | Setting |
|---|---|
| Enabled | If you want the timer to start working as soon as the form loads, set to true (.T.). Otherwise, leave this property set to false.) F.). You may choose to have an outside event (such as a click of a command button) start operation of the timer. |
| Interval | Number of milliseconds between timer events. |

وكمثال تطبيقي على Timer نأخذ مثال الساعة الإلكترونية كما موضح في الشكل (٥-٥).




**الشكل (٥-٥) مثال حول Timer**

ويبين الجدول التالي المسيطرات داخل النموذج أعلاه (ملاحظة:- Timer لا يظهر أثناء Run Time).

| Control | Property | Setting |
|---|---|---|
| LblTime | Caption | |
| Timer1 | Interval | ) ٥٠٠ half a second( |
| Timer1 | Enabled | True |

وتكتب العبارات البرمجية التالية في الحدث Timer.

```
IF THISFORM.lblTime.Caption != Time()
  THISFORM.lblTime.Caption = Time()
ENDIF
```

مع تغيير قيمة خاصية Interval إلى ٥٠٠ milliseconds (تقريبا ثانية واحدة).

## ي . استخدام Images

يسمح Images من إضافة صورة إلى النموذج (.bmp). وتمتلك Image معظم الخواص والصيغ والأحداث مثل بقية المسيطرات، فيمكن تغيير الصورة أثناء التنفيذ وعمل Click أو أي حدث آخر.

والجدول التالي يبين أهم خواص Image.

| Property | Description |
|---|---|
| Picture | The picture (.bmp file) to display. |
| BorderStyle | Whether there is a visible border for the image. |
| Stretch | If Stretch is set to 0 – Clip, portions of the picture that extend beyond the dimensions of the Image control are not displayed. If Stretch is set to 1 - Isometric, the Image control preserves the original dimensions of the picture and displays as much of the picture as the dimensions of the Image control will allow. If Stretch is set to 2 - Stretch, the picture is adjusted to exactly match the height and width of the Image control. |

## ك . استخدام Label

يختلف Label عن text box بما يلي:-

لا يوجد لها Data Source.

لا يمكن تحديث قيمتها بشكل مباشر.

لا يمكن الانتقال إليها عن طريق Tab.

يمكن تغيير خواص Caption و Visible برمجيا ليعكس ما هو مطلوب.

ويبين الجدول التالي أهم خصائص Label.

| Property | Description |
|---|---|
| Caption | The text displayed by the label. |
| AutoSize | Whether the size of the label is adjusted to the length of the Caption. |
| BackStyle | Whether the label is Opaque or Transparent. |
| WordWrap | Whether the text displayed on the label can wrap to additional lines. |

**ل . استخدام Shapes**

يمكن إضافة الأشكال الهندسية إلى النماذج لتحسين النموذج أو لتقسيم كائنات النموذج إلى مجاميع، ويبين الجدول التالي أهم خواص Shapes التي يمكن العامل معها في Design Time.

| Property | Description |
|---|---|
| Curvature | A value between 0 (90 degree angles) and 99 (circle or oval.( |
| FillStyle | Whether the shape is transparent or has a specified background fill pattern. |
| SpecialEffect | Whether the shape is plain or 3D. This only has an effect when the Curvature property is set to 0. |

**م . استخدام Line**

يستخدم Line بنفس استخدامات Shapes، ويبين الجدول التالي أهم خواص Line والتي يمكن التعامل معها أثناء Design Time.

| Property | Description |
|---|---|
| BorderWidth | How many pixels wide the line is. |
| LineSlant | When the line is not horizontal or vertical, the direction of the slant. Valid values for this property are a slash ( / ) and a backslash.( \ ) |

**ن . استخدام Grid**

هو عبارة عن Container، يحتوي على أعمدة Columns. وبالإضافة فإن Columns بدورها تحتوي على header وتحتوي على controls، كل واحدة لها خصائصها وأحداثها وصيغها مما يوفر سيطرة كبيرة على عناصر Grid.

Grid يوفر التعامل مع أعمدة وأسطر البيانات في النماذج، ولعل أفضل استخدام له هو في تصميم نماذج one-to-many مثل نماذج وصولات الاستلام والصرف في المخازن.

ولإضافة Grid إلى النموذج، يتم اختيار Grid من form control toolbar ومن ثم نسقطه على شاشة النموذج ونعمل له drug إلى الحجم المطلوب.

لتحديد عدد أعمدة grid نغير الخاصية ColumnCount إلى عدد الأعمدة المطلوب. وإذا عملنا قيمة هذه الخاصية مساوية إلى ١- فإن grid سوف يحتوي على عدد أعمدة مساوية لعدد الأعمدة في الجدول المرتبط به.

يمكن تحديث خواص أي عمود أو سطر في Grid وذلك إما عن طريق نقر الزر الأيمن للفأرة لإظهار قائمة الاختيارات الخاصة بالمسيطر ونختار منه Edit أو عن طريق نافذة Properties windows في مربع object نختار العمود المطلوب تحديث خواصه.

وعندما نكون في حالة edit للمسيطر grid فإنه سوف يرسم إطار حول grid، فإذا أردنا أن نخرج من Edit نضغط الفأرة في أي موقع آخر من النموذج.

ولتغير عرض العمود نحرك الفارة إلى الموقع بين عمودين في منطقة header فيتحول مؤشر الفأرة إلى خط عمودي وتظهر فيه أسهم صغيرة بالاتجاهين، نختار العمود ونعمل له drug إلى أن نصل إلى العرض المطلوب. أو يمكن عمل ذلك عن طريق تغيير خاصية Width في Properties Window.

ولتغيير ارتفاع السطر، ندخل في حالة edit ونحرك الفأرة إلى الموقع بين أول وثاني سطر في grid في الجهة اليسرى من grid فيتحول المؤشر إلى خط أفقي وله أسهم صغيرة من أعلى وأسفل، نختار السطر ونعمل له drug إلى الارتفاع المطلوب. أو يمكن عمل ذلك عن طريق تغيير خاصية Column's Hieght في Properties Window.

<u>ملاحظة:-</u> **يمكن منع المستخدم من تغيير ارتفاع الأسطر في Grid عن طريق تغيير قيمة الخاصية AllowRowSizing إلى .F..**

ولتغيير مصدر بيانات grid (يمكن تغيير مصدر بيانات grid ولكل column على حدة أيضا) يتم اختيار grid ونختار الخاصية RecordSourceType والذي يمكن تغييره إلى ٠ - Table إذا أردنا أن نفتح جدول ليرتبط به grid أو نجعل قيمته ١ - Alias إذا أردنا أن نربطه بجدول مفتوح أصلا. وبعدها نختار الخاصية RecordSource ونكتب اسم الجدول أو Alias ليكون هو مصدر بيانات grid. وإذا أردنا أن نحدد الحقل الذي يرتبط بكل عمود نغير خاصية datasource للعمود المطلوب.

ويمكن السماح للمستخدم من إضافة قيود جديدة إلى الجدول المعروض عن طريق grid عن طريق جعل الخاصية AllowAddNew مساوية إلى .T.، وللسيطرة الأكثر على الإدخال نجعل قيمة AllowAddNew مساوية إلى .F. ونستخدم بدلا عنه إيعاز Append Blank وإيعاز Insert.

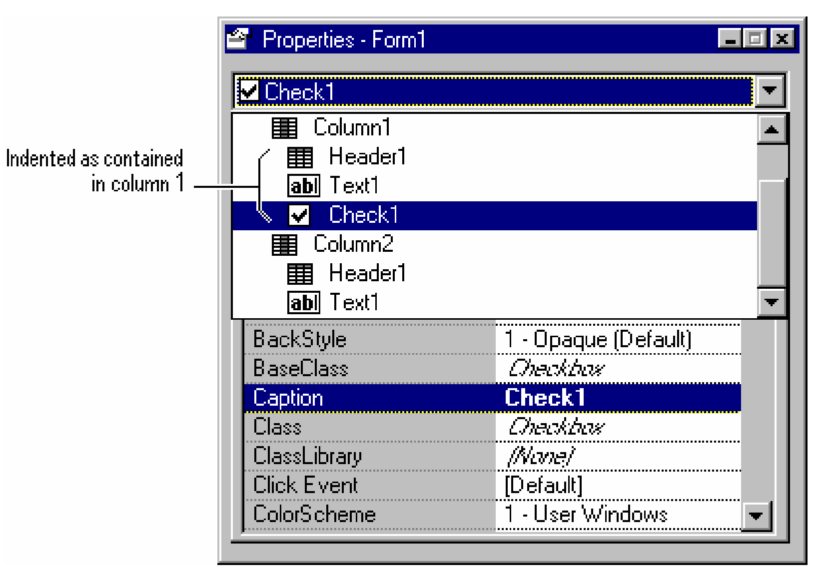
واحدة من أكثر استخدامات grid هو في عرض قيود الجدول الابن بينما نستعرض قيود الجدول الأب عن طريق المسيطرات الأخرى، بحيث إذا تنقل المستخدم عبر قيود الجدول الأب فإن القيود التي ستظهر في grid هي قيود الجدول الابن المقترنة بالجدول الأب.

إذا كانت العلاقة بين الجدول الأب والجدول الابن موجودة في data environment الخاص بالنموذج فإن عرض علاقة one-to-many بين الجدولين ستكون سهلة جدا.

ولغرض عرض control آخر (مثل combo box، check box، ...الخ) في أعمدة grid غير text box نتبع الخطوات التالية:-

في Properties Window نختار العمود المطلوب إضافة control إليه، فيظهر إطار حول grid للدلالة على حالة Edit.

من form controls toolbar نختار المسيطر المطلوب إضافته، وبعدها نضغط على العمود




**الشكل (٥-٦) مثال على إضافة control إلى عمود في grid**

المطلوب. المسيطر الجديد سوف لن يظهر في form designer ولكنه يكون visible في run time.
كما في الشكل (٥-٦).

نعمل خاصية sparse بالنسبة إلى العمود مساوية إلى .F..
نعمل خاصية CurrentControl بالنسبة للعمود إلى المسيطر الجديد الذي تمت إضافته.
وإذا أردنا أن نحذف المسيطر الإضافي داخل عمود grid نتبع الخطوات التالية:-
في مربع object الموجود في Properties Windows نختار المسيطر.
نجعل شاشة النموذج في حالة Active عن طريق ضغط الفأرة عليه.
نضغط مفتاح delete.

<u>**ملاحظة:-**</u> إذا كان المسيطر المضاف هو Combo Box ولغرض إظهاره بأفضل صورة نغير الخصائص التالية بالنسبة إلى Combo Box:-

BackStyle = 0
Margin = 0
SpecialEffect = 1
BorderStyle = 0

ويبين الجدول التالي أهم خصائص Grid:-

| Property | Description |
|---|---|
| ChildOrder | The foreign key of the child table that is joined with the primary key of the parent table. |
| ColumnCount | Number of columns. If ColumnCount is set to - 1, the grid has as many columns as there are fields in the grid's RecordSource. |
| LinkMaster | The parent table for child records displayed in the grid. |
| RecordSource | The data to be displayed in the grid. |
| RecordSourceType | Where the data displayed in the grid comes from:<br>a table، an alias، a query، or a table selected by the user in response to a prompt. |

ويبين الجدول التالي أهم خصائص الأعمدة:-

| Property | Description |
|---|---|
| ControlSource | The data to be displayed in the column. This is often a field in a table. |
| Sparse | If Sparse is set to true (.T.), controls in a grid are displayed as controls only when the cell in the column is selected. Other cells in the column display the underlying data value in a text box. Setting Sparse to true (.T.) allows faster repainting if a user is scrolling through a grid with a lot of displayed rows. |
| CurrentControl | Which control in the grid is active. The default is Text1, but if you add a control to the column, you can specify it as the CurrentControl. |

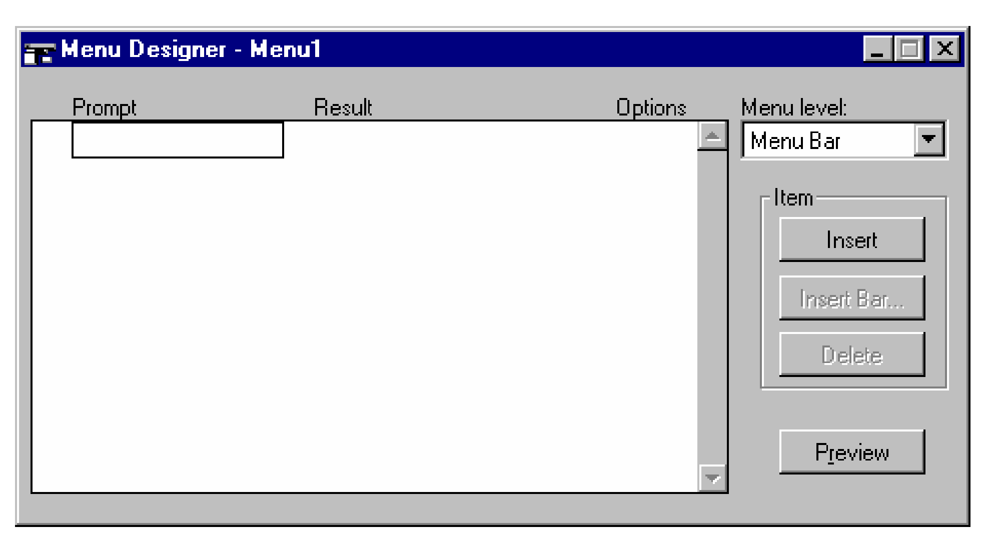
## ٦. تصميم القوائم وأشرطة الأدوات Menus And Toolbars design

القوائم (Menus) وأشرطة الأدوات (Toolbars) يقدمان طريق معالجة كفوء للمستخدم في السيطرة على إيعازات التطبيق. التصميم المناسب للقوائم وأشرطة الأدوات يوفران بيئة مناسبة للمستخدم بحيث لا يتضايق أو يمل عند استخدام التطبيق، ويفهم أسلوب التعامل مع التطبيق بسهولة ويسر.

### أ . إنشاء القوائم

معظم العمل الذي يتضمنه إنشاء القوائم يجري في مصمم القوائم (Menu Designer)، حيث يتم إنشاء القوائم، القوائم الفرعية واختيارات القوائم.

ويبين الشكل (٦-١) شاشة مصمم القوائم (menu designer).




**الشكل (٦-١) شاشة مصمم القوائم (Menu Designer)**

من قائمة File نختار New ونختار Menu، فتظهر لنا شاشة Menu Designer.

يتألف menu designer من ثلاثة أقسام رئيسية وهي:-

اولا . List box تحتوي على الخيارات التالية:-

(١). Prompt : يدرج فيه عنوان menu أو menu Bar.

(٢). Result : يظهر فيه combo box فيه أربعة عناصر وهي :-

(أ). Command :- إذا تم اختيار هذا العنصر فيكتب في text box المقابل الإيعاز المقترن بهذا الاختيار.

(ب). Pad name :- إذا تم اختيار هذا العنصر فيكتب في text box أسم pad الذي سيظهر في برنامج القائمة (.mpr) بعد عمل Generate له.

(ج). Submenu :- إذا تم اختيار هذا العنصر فسيتحول text box المقابل إلى button أسمه create، وعند الضغط عليه نتحول إلى شاشة تصميم القائمة الفرعية (ويتغير combo box في الجهة العلوية اليمنى من menu bar إلى اسم الاختيار الحالي المراد عمل submenu له).
(د). option:- هو عبارة عن button عند الضغط عليه تظهر شاشة اسمها Prompt option.

ثانيا . combo box اسمها menu level تؤشر فيها مستوى menu الحالي (menu bar أو اختيارات menu).

ثالثا . أربعة مسيطرات من نوع command buttons، وهي كالآتي:-
(١). Insert :- لإضافة اختيار جديد إلى menu.
(٢). Insert Bar :- لإضافة اختيار جديد إلى Submenu.
(٣). Delete :- لحذف الاختيار من menu و submenu.
(٤). Preview :- لعرض menu أثناء التصميم.

**ملاحظة:-** العلامة \< قبل أي حرف من اسم الاختيار يجعل تحت الحرف خط، ويمكن عمل activation لهذا الاختيار عن طريق ضغط هذا الحرف. والعلامة \- تفصل menu items عن بعضها عن طريق رسم خط.
قبل استخدام menu في التطبيق يجب عمل generation له، وذلك عن طريق اختيار قائمة menu وثم اختيار generate، فتظهر شاشة Prompt لخزن هذا menu (بأي اسم يتم تحديده) ويخزن ملف التصميم بالاستطالة (.mnx)، هذا الملف هو عبارة عن جدول يخزن فيه المعلومات عن menu. وبعد خزن menu تظهر شاشة أخرى Prompt لعمل output إلى ملف بنفس اسم menu ولكن بالاستطالة (.mpr)، ويحتوي هذا الملف على برنامج توليد menu.

**ب . إنشاء shortcut menus**

تظهر shortcut menus عند ضغط الزر الأيمن للفأرة على object، ويوفر طريقة سريعة لعرض الإجراءات والفعاليات التي تتعلق بهذا object فقط. نستخدم visual foxpro لإنشاء مثل هذه menus وبعدها نربطها مع object المطلوب. فمثلا يمكن عمل shortcut menu تحتوي عل الإيعازات cut، copy، paste والتي تظهر عند ضغط الزر الأيمن للفأرة على عمود في grid. ولإنشاء مثل هذه menus نختار القائمة file وبعدها new ثم menu ونختار shortcut. عملية إضافة menu items في shortcut menu هي نفسها في menu.
ولإضافة shortcut menu إلى object، نختار object، وندخل إلى الصيغة المتعلقة بالحدث right click، ونكتب الإيعاز التالي:-

```
do menu_name.mpr
```

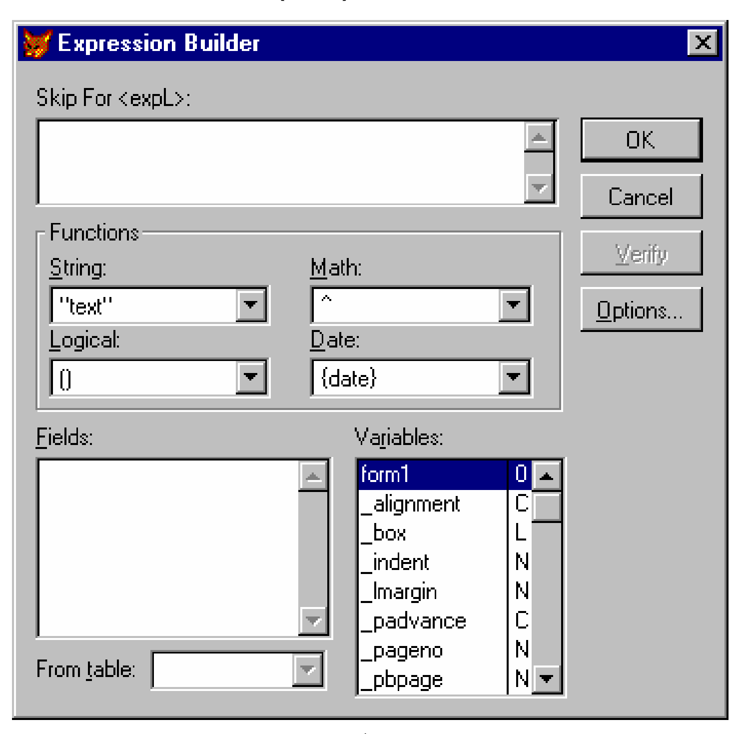
حيث أن menu_name هو اسم menu shortcut المطلوب، ويجب التأكد من إضافة extension (mpr.) إلى الاسم.

يمكن إضافة keyboard shortcut إلى اختيارات القوائم، وهي طريقة ضغط مفتاح معين بينما نكون مستمرين بالضغط على مفتاح آخر، والفرق بين حرف المعالجة (الحرف الذي يكون تحته خط) وبين keyboard shortcut هو أنه في keyboard shortcut نعالج الاختيار المطلوب دون الحاجة إلى إظهار القائمة المرتبط بها هذا الاختيار.

ولإنشاء menu shortcut نختار عنصر menu المطلوب ونضغط في حقل option الخاص به فتظهر شاشة prompt option. وفي مربع key label نضغط مجموعة المفاتيح المطلوبة لإنشاء هذا shortcut. وفي مربع key text نكتب العبارة التي نريد إظهارها جنب اختيار القائمة (عادة ما تكون هي نفسها shortcut).

**<u>ملاحظة:-</u> CTRL+J هو keyboard shortcut غير مسموح باستخدامه لأنه يغلق dialog box في VFP.**

ولغرض إيقاف عمل menu item معين إلا بشرط محدد يمكن استخدام مربع skip for الموجود في شاشة prompt option بعد ضغط option لاختيار القائمة المطلوب، فتظهر لنا شاشة expression builder. كما في الشكل (٦-٢)



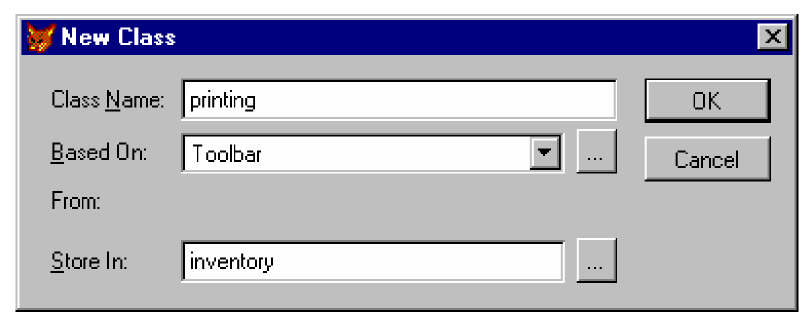

**الشكل (٦-٢) شاشة بناء التعابير**

إذا كان التعبير نتيجته (.F.) فإن اختيار القائمة يحصل له Enabled، أما إذا كان (.T.) فإن اختيار القائمة يحصل له Disabled.

## <u>ج. إنشاء أشرطة الأدوات Creating Toolbars</u>

إذا كان التطبيق يحتوي على فعاليات قد تتكرر أثناء العمل يمكن إضافة شريط أدوات لعمل هذه الفعاليات، ولغرض إنشاء toolbar يجب استخدام class خاصة بهذا toolbar.

نختار قائمة file وثم new ونختار class. في مربع class name نختار اسما لهذا class. ومن مربع Based On نختار Toolbar ليكون هو Class الأب. وفي مربع Store In نطبع المكتبة (Library) المراد خزن Class الجديد فيه، أو نختار Combo لاختيار مكتبة موجودة أصلا كما في الشكل (٦-٣). فتظهر لنا شاشة تصميم يمكن درج Object فيها وكتابة methods للأحداث التي تقع على هذا Object.



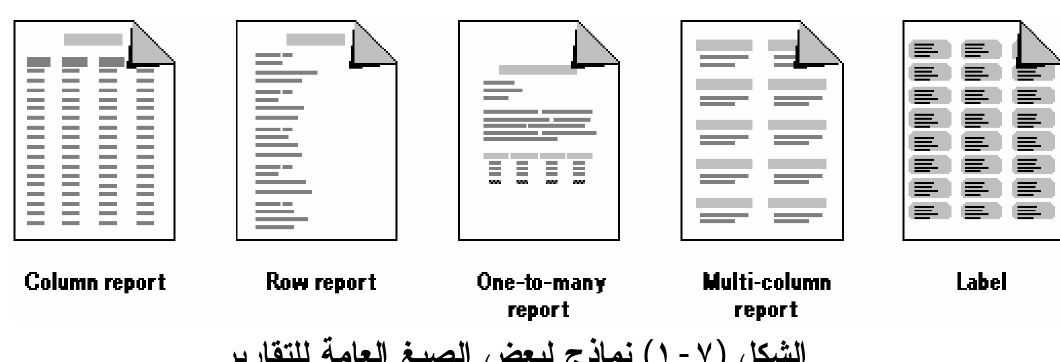

**الشكل (٦-٣) شاشة تحديد صنف جديد**

## ٧ . تصميم التقارير Reports Designing

هناك عدة طرق تستخدم لتصميم التقرير. وباستخدام مصمم التقارير report designer أو مـــنجم التقـــارير report wizard نستطيع إنشاء تقارير كفوءة وذات مظهر جيد وبأساليب عرض بيانات مختلفة. وفيما يلي الخطـــوط العامـــة لانشاء أي تقرير ترغب بتصميمه .

**أ . تحديد الخطوط العامة للتقرير**

قبل إنشاء التقرير نحدد الصيغة العامة للتقرير. فقد يكون التقرير بسيط ويشمل جدول واحد أو يكون التقرير معقـــدا ويحتوي على أكثر من جدول. ويبين الشكل (٧-١) نماذج لبعض الصيغ العامة للتقارير.




**الشكل (٧-١) نماذج لبعض الصيغ العامة للتقارير**

الجدول التالي يوضح هذه النماذج أعلاه.

| Layout Type | Description | Examples |
|---|---|---|
| Column | One record per row with fields placed horizontally across the page | Group/Total report[1]<br>Financial reports<br>Inventory<br>Sales summary |
| Row | One column of records with fields placed vertically down the side | Lists |
| One-to-many | One record or one-to-many relationship | Invoices<br>Account statements |
| Multi-column | More than one column of records with fields placed vertically down the left margin | Telephone directory<br>Business cards |
| Label | More than one column of records with fields placed vertically down the left margin; printed on special paper | Mailing labels[1]<br>Name tags |

**ب. إنشاء التقارير**

يمكن إنشاء التقارير في VFP بثلاثة طرق:-

اولا . إنشاء التقرير باستخدام Report Wizard.

ثانيا . إنشاء التقارير بشكل سريع عن طريق Quick Report.

ثالثا . تحديث تقرير موجود أو إنشاء تقرير عن طريق Report Designer.

## ج . استخدام Report Wizard

يمكن استخدام report wizard في تصميم التقارير ومن ثم تعديل التصميم في report designer حيث يوفر report wizard الفقرات التالية:-

.Report
One-to-Many
Label
Mail Merge

**ملاحظة:- (الفقرتين الأخيرتين غير موجودة في نسخ تطبيق VFP المتداولة حاليا).**

ولتصميم التقرير نختار من قائمة File اختيار New ومن ثم Report. وبعدها نختار Report Wizard.

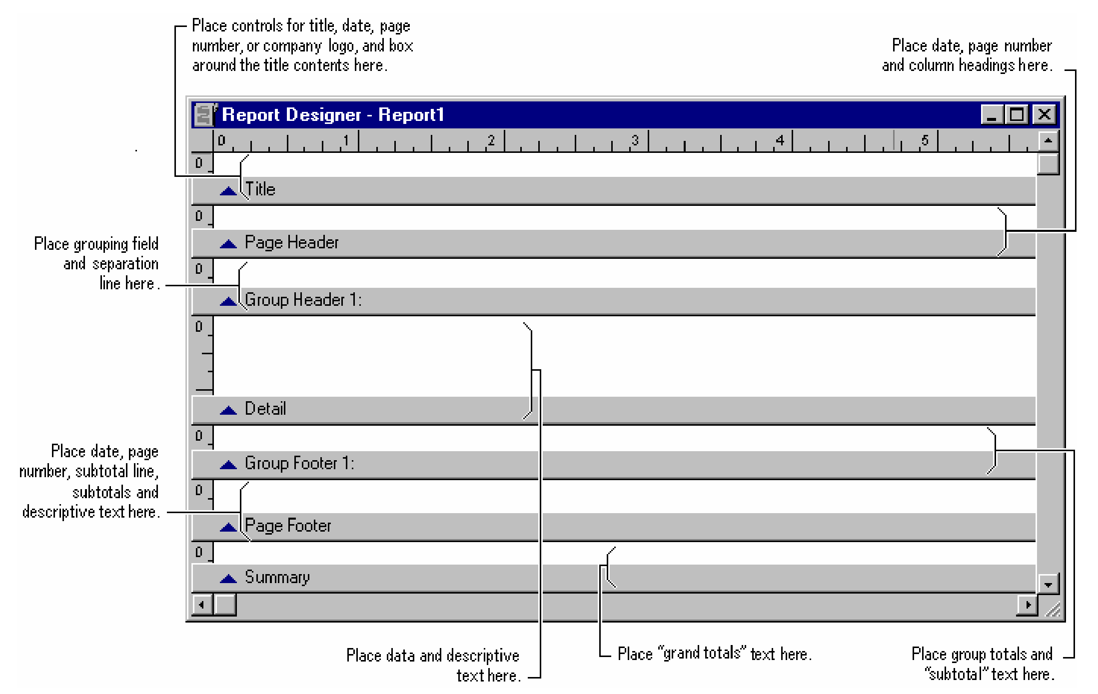
كما ويمكن اختيار report wizard من قائمة tools واختيار report،وبنفس الطريقة يمكن اختيار Label و Mail.

## د . تحديث محتويات التقارير

إذا كان لدينا تقرير فارغ (مصمم عن طريق report designer)، أو wizard مصمم عن طريق report wizard أو quick report، فيمكن تحديث محتويات التقرير عن طريق report designer.

في حزم (bands) داخل report designer يمكن إضافة controls التي تسيطر على العناوين labels، الحقول، والمتغيرات، وكذلك التعبيرات التي نرغب في طباعتها في التقرير. ولتحسين مظهر التقرير يمكن إدراج الخطوط والأشكال الهندسية (مربعات، دوائر...الخ)، وإدراج الصور والألوان.

يتم استخدام ما يسمى (bands) لتحديد محتويات كل صفحة، مجموعة، وبداية ونهاية التقرير. يمكن تغيير حجم bands وإضافة controls، ونقلها من مكان إلى آخر وترتيب مواضعها في التقرير. والتقرير يحتوي على عدد من bands، ويبين الشكل (٧-٢) بعض أنواع bands التي يمكن إدراجها في report.




**الشكل (٧-٢) بعض أنواع bands في التقارير**

ويمكن أن يحتوي التقرير على عدد من الحزم (goup bands) ويبين الجدول التالي أنواع استخدامات bands.

| Use this band | To print | Use this command |
|---|---|---|
| Title | Once per report | Choose **Title/Summary** from the **Report** menu. |
| Page header | Once per page | Available by default. |
| Column header | Once per column | Choose **Page Setup** from the **File** menu and set **Column Number** greater than 1. |
| Group header | Once per group | Choose **Data Grouping** from the **Report** menu. |
| Detail band | Once per record | Available by default. |
| Group footer | Once per group | Choose **Data Grouping** from the **Report** menu. |
| Column footer | Once per column | Choose **Page Setup** from the **File** menu and set **Column Number** greater than 1. |
| Page footer | Once per page | Available by default. |
| Summary | Once per report | Choose **Title/Summary** from the **Report** menu. |

## هـ . إضافة مسيطرات التقرير (Report Control)

يبين الجدول التالي أنواع Controls التي يمكن إضافتها إلى التقرير.

| To display | Choose this control |
|---|---|
| Table fields, variables, and other expressions | Field |
| Literal text | Label |
| Straight lines | Line |
| Boxes and borders | Rectangle |
| Circles, ovals, boxes with rounded corners, and borders | Rounded Rectangle |
| Bitmaps or general fields | Picture/OLE Bound |

## و . تحديد مصادر بيانات التقرير

يمكن إضافة مصادر البيانات إلى التقارير بسهولة عن طريق data environment الخاصة بالتقرير، حيث يمكن إضافة الجداول والمنظورات وترتيب هذه الجداول والمنظورات وفهرستها وحسب متطلبات التقرير.

وتقوم Data environment بإدارة مصادر البيانات في التقرير على النحو التالي:-

فتح الجداول والمنظورات عند فتح أو تنفيذ التقارير.

التعامل مع البيانات المقترنة بالجداول والمنظورات.

غلق الجداول عند انتهاء التقرير.

ولإضافة جدول أو منظور إلى data environment، من قائمة view نختار data environment أو عن طريق نقر الزر الأيمن للفأرة ومن ثم نختار Add.

ويتم التعامل مع data environment بنفس الطريقة عند تصميم النماذج من حيث الفهرسة والعلاقات...الخ.

**ز . إضافة مسيطرات الحقول**

يمكن ان يحتوي التقرير على field controls والتي تمثل قيم الحقول في جدول أو منظور، قيم المتغيرات، أو القيم الناتجة عن حسابات معينة.

ولإضافة حقل موجود في جدول ضمن data environment الخاصة بالتقرير، نقوم أولا بالدخول إلى شاشة data environment ومن ثم نؤشر الجدول المطلوب والحقل المراد درجه في التقرير ونعمل له drug إلى أن نصل بمؤشر الفأرة إلى الموقع المطلوب في مساحة التقرير ونعمل له drop في ذلك الموقع.

ويمكن إضافة هذا الحقل عن طريق report designer toolbar أيضا.

ولعمل هذا نتبع الخطوات التالية:-

من report control toolbar نضيف مسيطر field.

بعد لإضافة المسيطر تظهر لنا شاشة report expression فنختار button بعد مربع expression.

في مربع field نعمل double click للحقل المطلوب.

**ح . إضافة Label Control**

يمكن إضافة Label لطبع التعبيرات الثابتة في التقرير مثل العناوين، عناوين الحقول، أو أي معلومات أخرى.

ولإضافة مسيطر label نتبع الخطوات التالية:-

من report controls toolbar نختار label.

من report designer نضغط على المسيطر ثم نؤشر موقعه في التقرير ونكتب العبارة المطلوبة.

ولتحديث label موجود نؤشر على المسيطر label في report control toolbar ومن ثم نؤشر على label المراد تحديثه ونجري عليه التحديث.

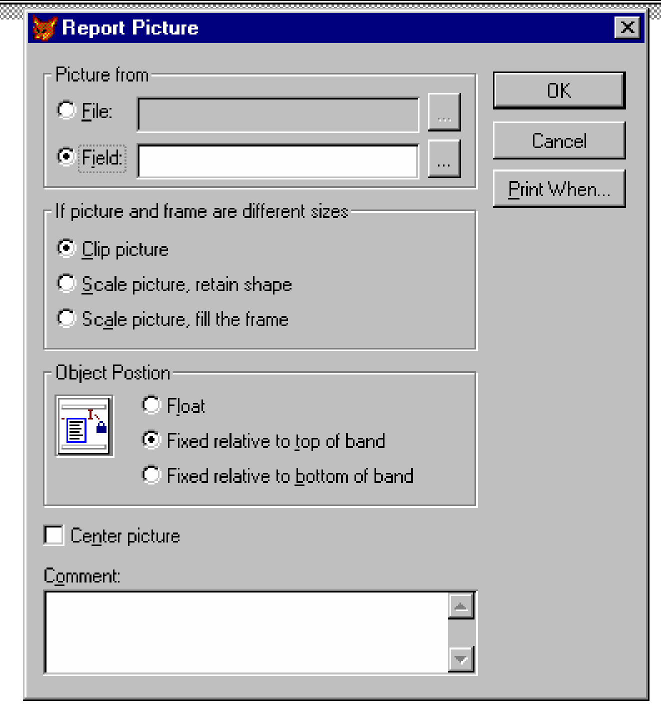
**ط . إضافة حقل عام (General Field)**

يمكن إضافة حقل عام يحتوي على OLE في التقرير عن طريق ما يلي:-

من report control toolbar نؤشر Picture/OLE Bound.

نعمل له drug في الموقع المطلوب في التقرير فتظهر لنا الشاشة الموضحة بالشكل (٧-٣) وهي شاشة report picture.

في مربع field نطبع اسم الملف أو نضغط button في نهاية مربع field لاختيار الملف المطلوب.

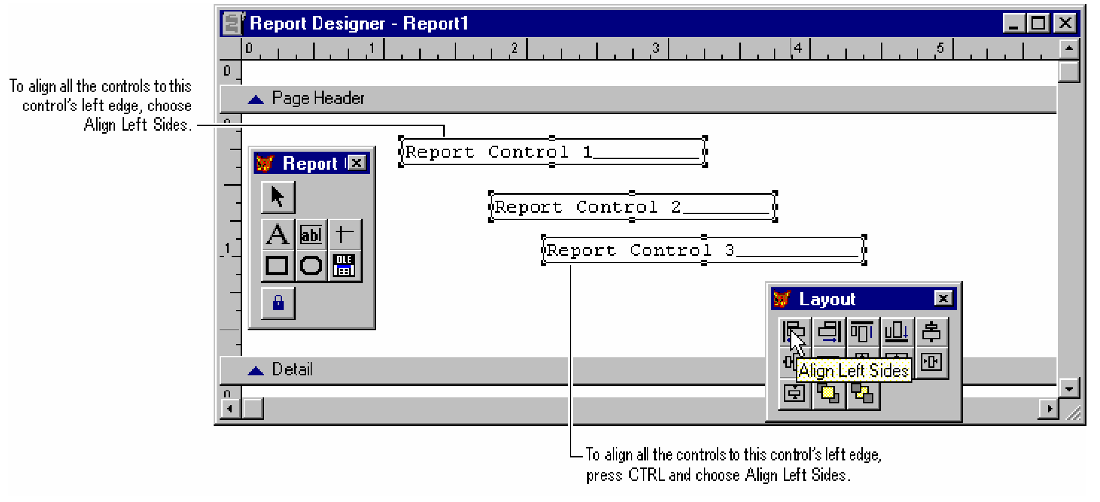



**الشكل (٧-٣) شاشة report picture**

**ملاحظة:-** يمكن عمل copy، cut، و paste لأي control في report، حيث يمكن التعامل مع كائنات report مثل أي تطبيق آخر من تطبيقات Windows.

## ي . ترتيب المسيطرات في التقرير

نستطيع عمل ترتيب للمسيطرات (Alignment) لكل مسيطر على حدة أو بالنسبة إلى مواقع المسيطرات الأخرى ويمكن عمل ذلك عن طريق toolbar layout. كما موضح في الشكل (٧-٤). حيث يمكن نقل control إلى اليسار أو اليمين، توسيط control إظهاره أمام control آخر أو إخفاءه خلفه.




**الشكل (٧-٤) Layout Toolbar**

**ك . تجميع البيانات Grouping Data**

بعد الانتهاء من تصميم الخطوط العامة للتقرير، نبدأ بتحديد مجموعات البيانات في التقرير. تتيح هذه العملية فصل بيانات كل مجموعة عن الأخرى وإظهار العناوين والخلاصات لكل مجموعة.

عند عمل مجموعات، فإن كل مجموعة سوف تمتلك Group Header و Footer حيث يمكن إضافة مسيطرات له.

كما ويمكن أيضا تحديد خيارات أخرى لكل مجموعة وهي:-

طبع العبارات في header و Footer لتعريف كل مجموعة.

طبع كل مجموعة في صفحة منفصلة.

إعادة تضفير عداد الصفحات عند طبع كل مجموعة في صفحة جديدة.

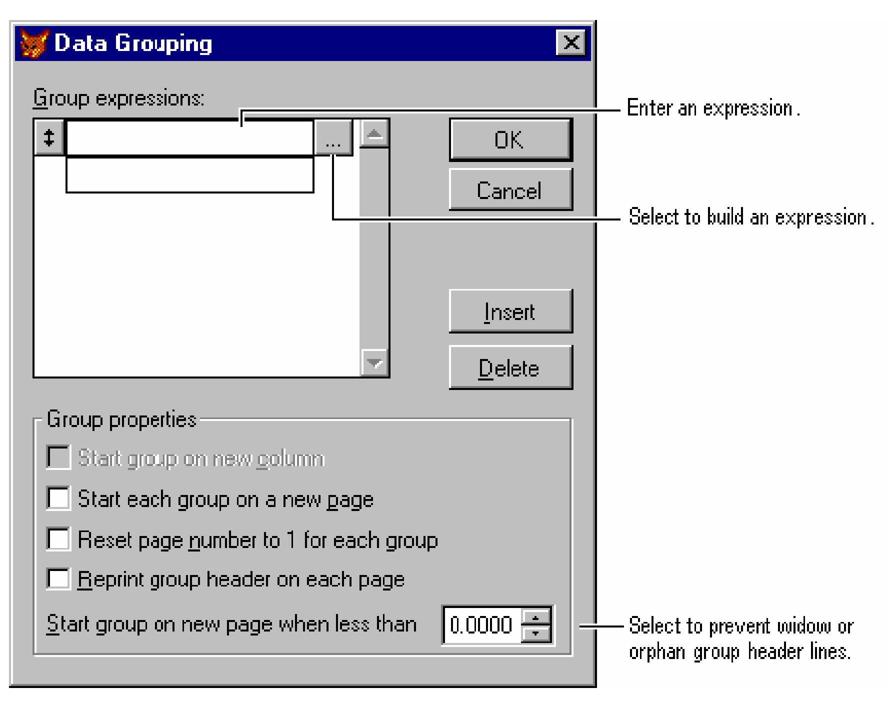
كما ونستطيع إظهار البيانات وفق الترتيب المطلوب عن طريق فتح الجدول في بيئة بيانات التقرير على الفهرسة المطلوبة.

**ل . إضافة مجموعة مفردة**

التقرير ذو المجموعة المنفردة هو عبارة عن مستوى تجميع بيانات واحد، فمثلا إذا أردنا أن نطبع تقرير لطبع كل القيود لأفراد يعيشون في محافظة واحدة فنضع group لكل محافظة (بشرط عمل فهرسة للجدول على المحافظة).

ولإضافة مجموعة إلى التقرير نتبع ما يلي:-

من قائمة Report نختار Data Grouping، فتظهر لنا شاشة Data Grouping Dialog box كما في الشكل (٧-٥).




**الشكل (٧-٥) Data Grouping Dialog Box**

في مربع Group نطبع التعبير المطلوب. أو نضغط Button للدخول إلى شاشة Expression Builder.

في المساحة Group Properties نحدد خيارات group.

بعد إنشاء expression ننقل أحد controls إلى bands، وعادة يكون control المستخدم لعمل grouping، حيث يتم نقله من Detail Band إلى Group Header Band.

## م . إضافة Data Grouping متعددة

يمكن عمل ٢٠ مجموعة (data grouping) لكل تقرير .

و لإضافة multiple group نتبع الخطوات التالية:-

من قائمة report نختار Data Grouping الشكل (٧-٥).

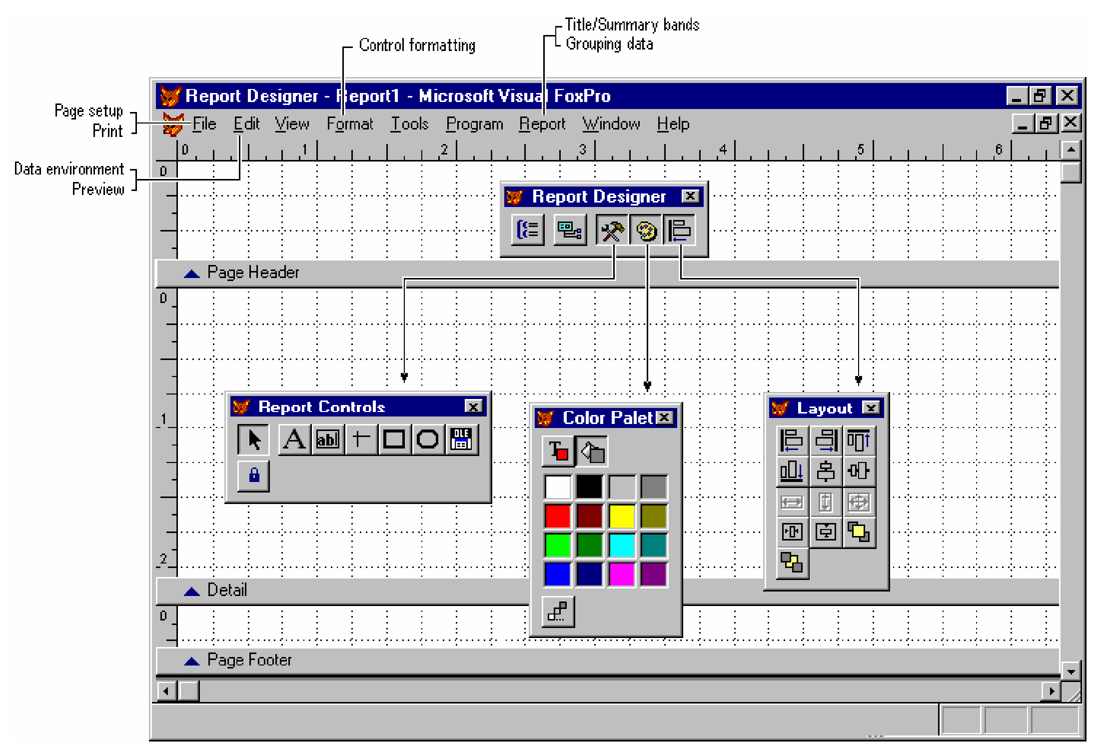
في مربع أول Group نطبع expression أو نضغط button للدخول في expression builder.

في Group Properties نختار Properties المطلوبة.

نختار Insert ونعيد الخطوتين السابقتين لكل group.

## ن . تحسين مظهر التقرير

بعد الانتهاء من تحديد الخطوط العامة وبيئة البيانات للتقرير وإنشاء المجموعات، يمكن تحسين المظهـر الخـارجي للتقرير عن طريق استخدام tools المتوفرة لهذا الغرض ويبين الشكل (٧-٦) نموذج لتقرير مع الإمكانيات المتاحـة لتحسين المظهر الخارجي له.

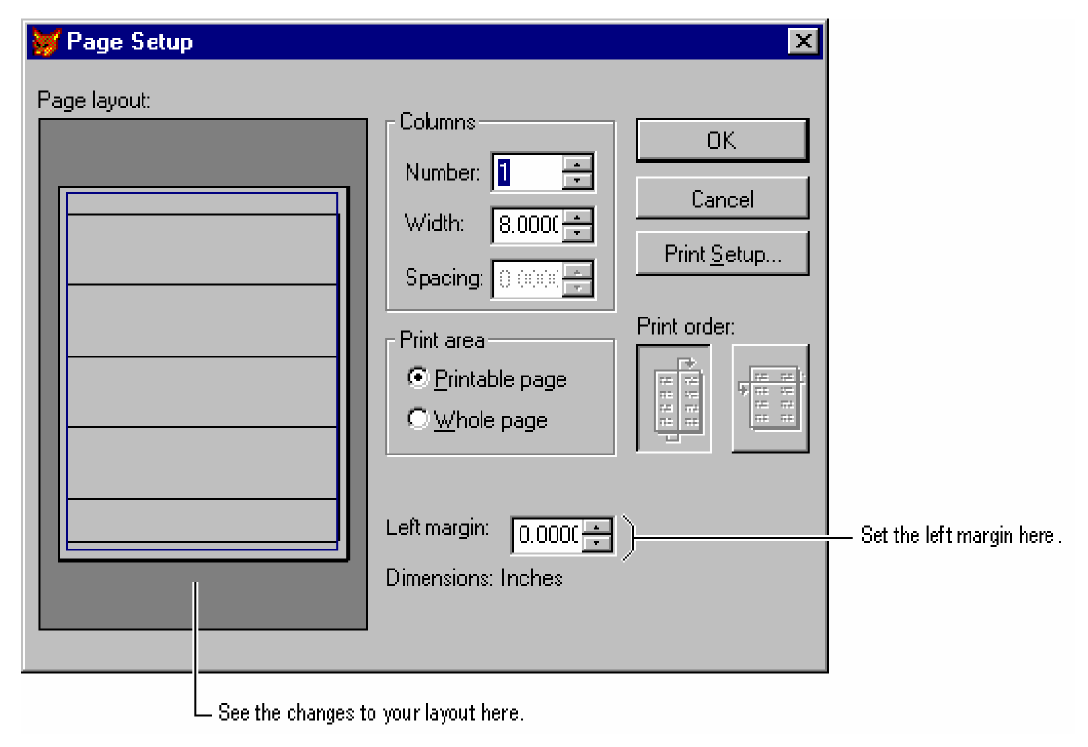



الشكل (٧-٦) Tools المتاحة لتحسين مظهر التقرير

**س . تعريف صفحة التقرير**

عند التخطيط لإنشاء التقرير يجب مسبقا معرفة كيفية ظهور الصفحة من حيث الحواشي، نوع الورقة وحجمها وارتفاع كل حزمة (band) وتفاصيل المسيطرات في التقرير.

ولتحديد الهامش الأيسر للتقرير، نختار من قائمة File اختيار Page Setup. فتظهر لنا شاشة Page Setup Dialog Box كما في الشكل (٧-٧).



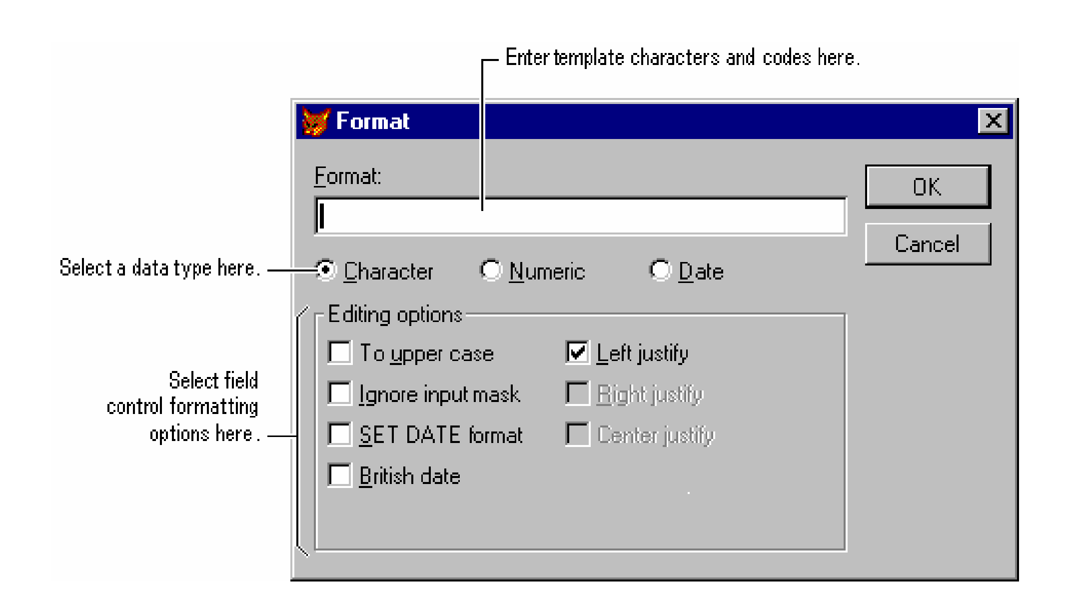

**الشكل (٧-٧) شاشة Page Setup**

ولتحديد الهامش، ندخل الرقم المطلوب للهامش في مربع Left Margin، فتتغير صورة الصفحة لإظهار الهامش الجديد. وفي مربع Print Setup نختار حجم الورقة المطلوب من قائمة Size. ولاختيار اتجاه الورقة (landscape أو Portrait) نضغط على Print Setup ونختار الخيار المطلوب من مساحة Orientation.

**ع . تحديد صيغة الحقول والمسيطرات**

بعد إدراج الحقول والمسيطرات يمكن تغيير صيغتها ونوع البيانات التي يمكن طبعها، حيث إن البيانات قد تكون Character، Numeric، أو Date. وكل نوع من هذه الأنواع لها الصيغ الخاصة بها.

ولتحديد صيغة Field نضغط Double Click على Field. فتظهر لنا Report Expression Dialog Box فنضغط button بعد مربع Format. فتظهر لنا شاشة Format Dialog box كما في الشكل (٧-٨). ومن مربع Format نختار نوع البيانات لهذا الحقل أو المسيطر، فتظهر في مساحة Editing الخيارات المتاحة للصيغ تبعا لنوع البيانات. وبعدها نختار موقع الطبع وخيارات الصيغة.




**الشكل (٧-٨) شاشة تحديد صيغ ظهور البيانات في التقرير**

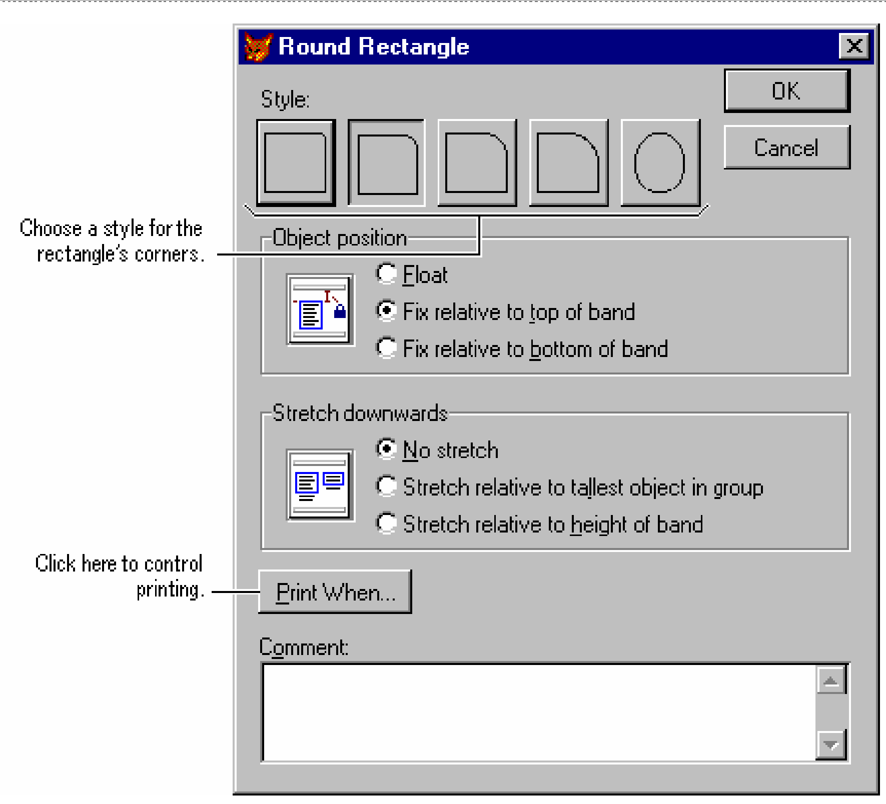
## ف. تغيير البنط Changing Fonts

يمكن تغيير حجم الخط ونوعه للحقول والعناوين وذلك عن طريق تأشير المسيطر المطلوب تغيير خطه، ومن قائمة Format نختار Font فتظهر لنا شاشة Font Dialog Box، فنختار منها نوع الخط وحجمه وطبيعة الخط (Bold، Italic،...الخ). ولتغيير Default Font للتقرير نختار من قائمة Report اختيار Default Font ومن ثم نختار نوع وحجم الخط المطلوب لأن يكون Default Font لهذا التقرير.

## ص. إضافة الخطوط والمستطيلات والدوائر

يمكن إضافة الخطوط والأشكال الهندسية إلى التقرير لفصل وترتيب أجزاء التقرير. ويمكن اختيار الأشكال الهندسية من report Controls toolbar، عن طريق اختيار أحد buttons الخاصة بالرسم.

عند اختيار المستطيل أو الدائرة، فإننا وبمجرد عمل Double Click عليه نستطيع تغيير خوا الشكل بعد ظهور شاشة Rectangular Round كما في الشكل (٧-٩) عند الضغط على circle أو شاشة Rectangular/Line عند الضغط على مستطيل أو خط.




**الشكل (٧ - ٩) شاشة Round Rectangular**

**ق . تغيير عرض ونوع الخطوط للأشكال**

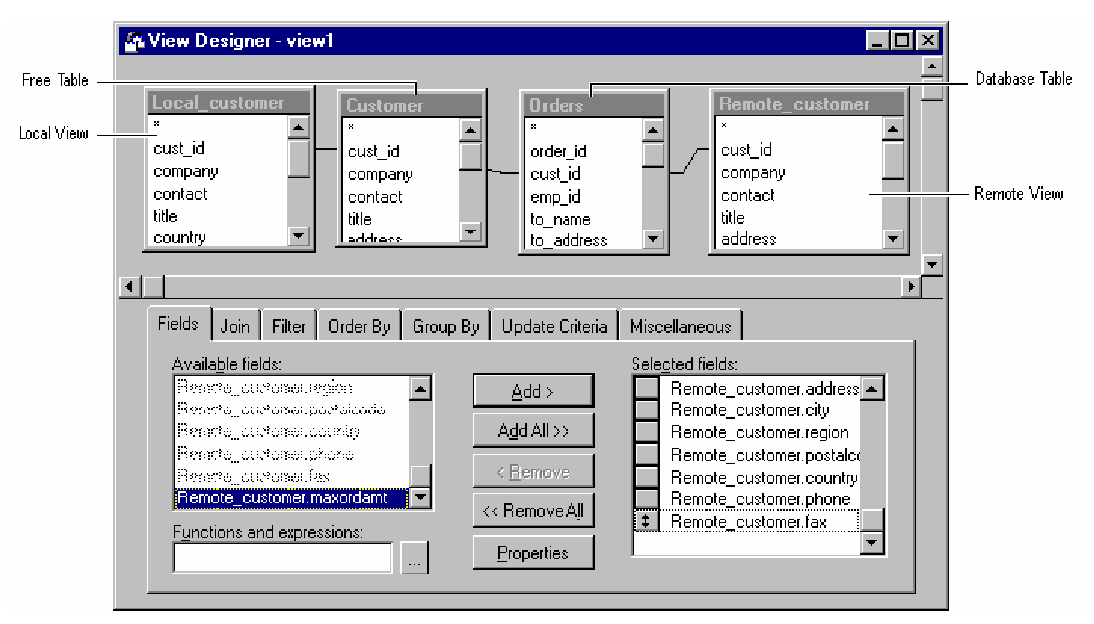
يمكن تغيير عرض الخط أو نوعه للأشكال الهندسية المختلفة في التقرير عن طريق قائمة Format واختيـــار Pen، ومن القائمة الفرعية التي تظهر نختار النوع والحجم المناسب للخط.

# ٨ . تصميم الاستفسارات والمنظورات Query and View Designing

عند استخدام VFP يمكن إنشاء الاستفسارات لعدد من الجداول والمنظورات، وتجميع البيانات ومن قواعد محلية أو قواعد بعيدة في منظورات داخل قاعدة البيانات. وإمكانية إنشاء الاستفسارات مفيدة جدا في أنظمة إدارة قواعد البيانات وخاصة إذا أردنا تجميع البيانات من أكثر من جدول عن طريق ترابط العلاقات بينها.

## أ . الاستفسار باستخدام جداول ومنظورات متعددة

إذا أردنا معالجة البيانات في أكثر من جدول أو منظور، فنقوم أولا بإضافتها إلى الاستفسار ومن ثم ربطها (join) عن طريق الحقول المشتركة بينها. كما في الشكل (٨-١).

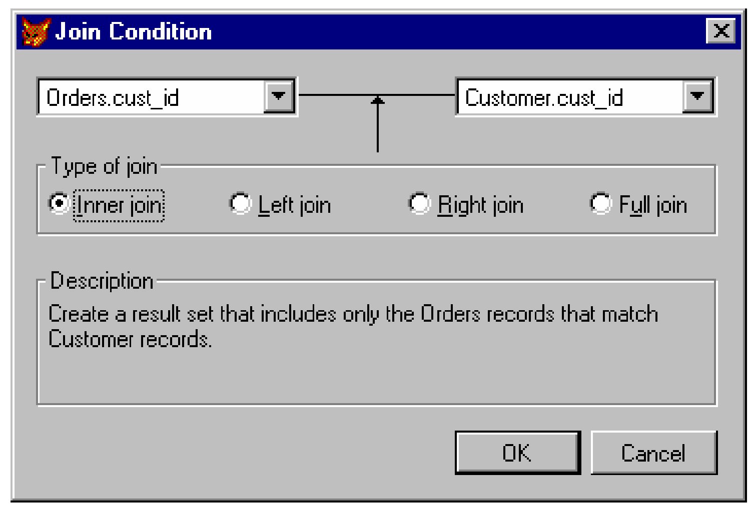



**الشكل (٨-١) شاشة مصمم المنظورات**

إذا استخدمنا قاعدة بيانات تحتوي على علاقات ثابتة بين الجداول فإن VFP سيستخدم هذه العلاقات كعلاقات ربط بديهية (Default Joins). ولإضافة جدول أو منظور إلى الاستفسار فيجب أولا فتح قاعدة البيانات ومن Query Designer Toolbar نختار Add Table. ومن مربع Add Table Or View نختار قاعدة البيانات المطلوبة ونختار الجدول أو المنظور المطلوب درجه في الاستفسار، أو نختار Other لتحديد جدول حر (free table).

## ب . السيطرة على اختيار القيود باستخدام Joins

عند استخدام جداول متعددة في الاستفسار فإن طريقة تحديد السيطرة على القيود واختيارها تتم من خلال استخدام joins. وباستخدام join condition dialog box يمكن تغيير نوع join بين جدولين، يظهر join بصورة أوتوماتيكية عند إضافة الجداول، كما ويمكن إنشاء joins عن طريق عمل drug بين الحقول في query designer، أو بالضغط على Add Join Button، كما في الشكل (٨-٢).




**الشكل (٨-٢) شاشة join condition**

وبعد إضافة join أو تحديث Join موجود (عن طريق عمل Double Click)، نستطيع اختيار نــوع join لتوســيع حجم النتائج أو تضيقها. وأسهل طريقة لإنشاء join هو عن طريق استخدام join condition dialog box.

ولإنشاء join بين جدولين نتبع ما يلي:-

نضيف الجداول إلى الاستفسار.

من query designer toolbar نختار Add Join.

من مربع join condition نختار حقول الربط (مع ملاحظة أن الحقول يجب أن تكون من نفس نوع البيانات ولها نفس الحجم).

بعدها نختار نوع join، والربط على عدة أنواع، ويبين الجدول التالي أنواع join بين جداول VFP.

| To retrieve… | Use… |
|---|---|
| Only records from both tables that match the join criteria, the most common type of join | Inner Join |
| All records from the table on the left side of the join criteria and only records that match the join criteria from the table on the right side of the join criteria | Left Join |
| All records from the table on the right side of the join criteria and only records that match the join criteria from the table on the left side of the join criteria | Right Join |
| All records from both tables whether or not they match the join criteria | Full Join |

## ج . استخدام جداول متعددة في منظورات

يمكن أيضا ربط جدولين أو أكثر من local tables من خلال local views. وطريقة المعالجة تشبه تمامـــا طريقــة معالجة الجداول المتعددة (الفقرة السابقة من المحاضرة). ولكن استخدام views تضيف لنا خاصية أخرى مهمة وهي خاصية إمكانية تحديث جدول مصدر بيانات view. أي أنه إذا أنشأنا view لجدول موجــود فـــي قاعــدة البيانـــات

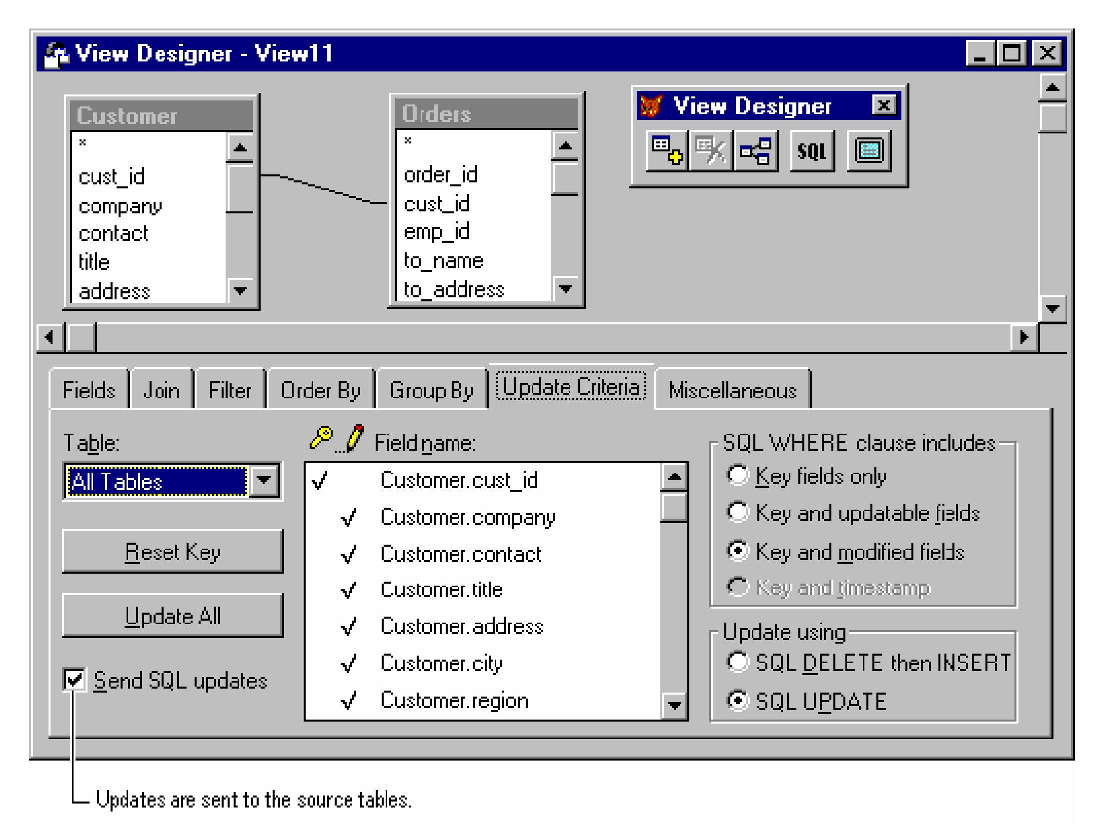
وتعاملنا مع view المتولدة في نموذج أو في شاشة استعراض بيانات (browse)، وقمنا بتحديث البيانات فإن البيانات في الجدول المصدر سوف تتغير أيضا.

ويبين الشكل (٨-٣) نموذج لربط جداول عن طري إنشاء View.

ولربط الجداول في view نختار Add Table في View Designer، ومن ثم نختار Add Join، نغير join conditions كما نشاء ونختار OK.

وإذا أردنا أن نعمل enable لتحديث مصدر البيانات نختار Update Criteria Tab، وبعدها نختار Send SQL updates.

<u>**ملاحظة:-**</u> لعمل activate إلى Send SQL updates يجب أولا تحديد أحد الحقول في أحد الجداول ليكون Key Field.




**الشكل (٨-٣) ربط الجداول باستخدام view**

## <u>د . إضافة Query باستخدام SQL-Statement</u>

عند إضافة Queries إلى التطبيق فإننا نستطيع جمع البيانات من مصادر بيانات متعددة ومختلفة، بالإضافة إلى عمل filter للبيانات، ترتيب البيانات، ...الخ، كل هذا باستخدام SELECT-SQL. وباستخدام هذا الإيعاز فإننا سنمتلك السيطرة الكاملة على البيانات الناتجة وعلى موقع إخراج هذه النتائج.

وفيما يلي بعض الأمثلة على استخدام SQL-Statement:-

```
SELECT ؛ *
FROM tastrade!customer؛
WHERE customer.country = "Iraq"
```

في هذا المثال يتم اختيار كل القيود من جدول اسمه customer موجود في قاعدة بيانات اسمها testdata للقيود التي فيها قيمة خاصية الدولة فيها مساوية إلى Iraq. ومما يجدر بالملاحظة في المثال أعلاه هو فصل اسم القاعدة عن اسم الجدول بعلامة (!).

```
SELECT TOP 10؛*
FROM testdata!customer INNER JOIN testdata!orders؛
ON Customer.cust_id = Orders.cust_id;
GROUP BY Customer.cust_id؛
ORDER BY Orders.order_amt DESC
```

في المثال أعلاه نقوم باختيار أول عشرة زبائن لجدول customer في قاعدة testdata مرتبين حسب أعلى قيمة طلب (حقل order_amt) الموجود في جدول الربط orders في نفس القاعدة، ونستخدم GROUP BY لحقل cust_id في جدول customer لإظهار طلب واحد لكل زبون فقط.

**هـ. تحديد وجهة الإخراج لنتائج الاستفسار**

عن طريق استخدام SQL-Statement يمكن تحديد وجهة الإخراج لنتائج الاستفسار، ويبين الجدول التالي أنواع الإخراج بالنسبة إلى queries.

| To send results to this destination | Use this clause |
|---|---|
| Separate table | INTO TABLE [table name[ |
| Array | INTO ARRAY [Array name[ |
| Temporary table | INTO CURSOR [cursor name[ |
| Active window | TO SCREEN |
| Browse window | The default if no other destination is specified. |

لتحديد نوع الإخراج بجدول نضيف INTO إلى SQL كما في المثال التالي:-

```
SELECT * ;
  FROM tastrade!customer ;
  WHERE customer.country = "Iraq" ;
  INTO TABLE mytable
```

والمثال التالي يبين وجهة الإخراج إلى Array:-

```
SELECT * ;
  FROM tastrade!customer ;
  WHERE customer.country = "Iraq" ;
  INTO ARRAY aMyArray
```

والمثال التالي يبين وجهة الإخراج إلى Cursor:-

```
SELECT * ;
  FROM tastrade!customer ;
  WHERE customer.country = "Iraq" ;
  INTO CURSOR mycursor
```

**و . طبع نتائج الاستفسارات باستخدام التقارير**

يبين المثال التالي كيفية إظهار نتائج الاستفسار في تقرير:-

```
SELECT * ;
FROM tastrade!customer ;
WHERE customer.country = "Iraq" ;
GROUP BY customer.region ;
ORDER BY customer.postal_code, customer.company_name ;
INTO CURSOR MyCursor
REPORT FORM MYREPORT.FRX
```

**ملاحظة:-** يمكن إنشاء Cross Tab query (يشبه spread sheet في Excel) وكذلك Graph Query باستخدام Query Wizard.

## ٩ . بناء التطبيقات Building Applications

يمكن بناء تطبيقات object-oriented و event-driven، على شكل عناصر منفصلة وكل على حدة مما يسهل عملية فحص هذه التطبيقات وتحسين أدائها. وبعد بناء كل هذه العناصر نستطيع جمعها بشكل موحد في تطبيق واحد، هذا التطبيق يجمع كل أجزاء التطبيق في تطبيق واحد مما يسهل علينا العمل في توزيع التطبيق على المستخدمين بشكل سهل و كفوء.

عادة ما يتألف تطبيق VFP من قائمة رئيسية للنظام (أو نموذج رئيسي للنظام) وعدد من النماذج لإدخال البيانات أو عرضها وعدد من التقارير التي تتيح للمستخدم استعادة البيانات بأشكال مختلفة، وبعض البرامج التي تضمن تكاملية البيانات.

وعند البدء في بناء هيكل التطبيق يجب ملاحظة النقاط التالية والتأكيد عليها:-

أ . تحديد نقطة البدء عند تنفيذ التطبيق.

ب . تحميل بيئة التطبيق (Environment).

ج . عرض واجهة التطبيق الرئيسية.

د . السيطرة على دورة الأحداث (Events Loop).

هـ . استرجاع البيئة الأساسية عند الخروج من التطبيق.

و . بناء التطبيق من مشروع.

### أ . تحديد نقطة البدء في التطبيق

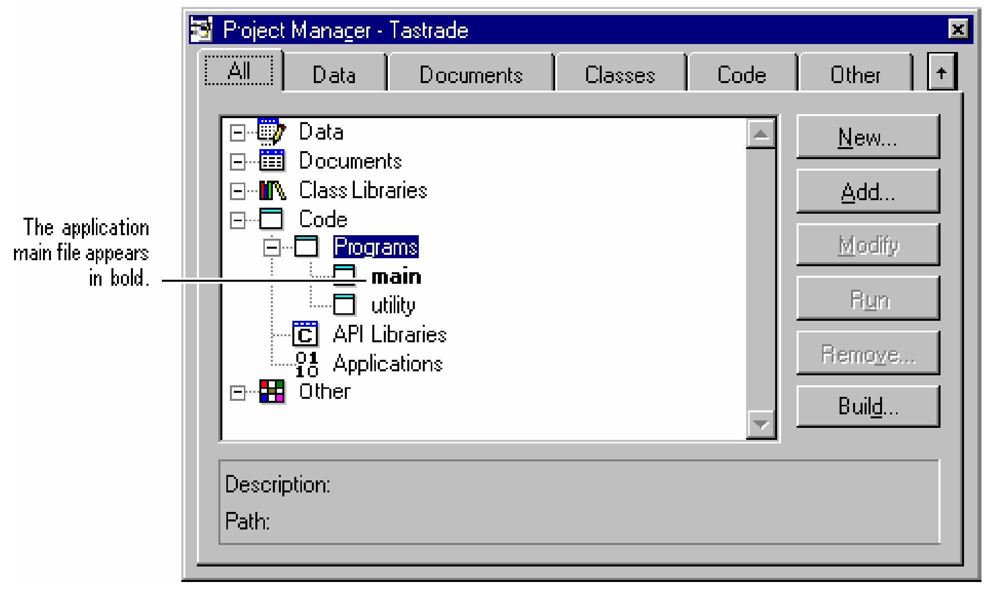
يتم ربط كل جزء من أجزاء التطبيق مع الجزء المتعلق به وتحدد النقطة الرئيسية من خلال تحديد الملف الرئيسي للتنفيذ (main file). هذا الملف الرئيسي يكون هو نقطة البدء عند تنفيذ التطبيق وقد يتكون من برنامج أو نموذج، وعندما ينفذ VFP التطبيق الخاص بنا فإنه يقوم بتحميل هذا الملف في البداية، ليقوم هذا الملف الرئيسي بتنفيذ بقية أجزاء التطبيق. وكل تطبيق يجب أن يحتوي على ملف رئيسي وعادة ما يكون عبارة عن برنامج، كما ويمكن استخدام نموذج كملف رئيسي يحتوي على عناصر البرامج الرئيسي ضمنيا. إذا تم بناء التطبيق باستخدام Application Wizard فسوف يتيح لنا أن نعرف الملف الرئيسي للتطبيق، كما ويمكن لنا أن نقوم بتسمية أو تغيير الملف الرئيسي وفقا لخصوصية ونوع التطبيق.

ولغرض تحديد نقطة البدء للتطبيق نتبع ما يلي:-

في مدير المشروع (Project Manger) نحدد الملف، كما في الشكل (٩-١).

من قائمة project نختار Set Main، أو عن طريق ضغط الزر الأيمن للفأرة على الملف المطلوب.

ملاحظة:- عند تحديد الملف الرئيسي فإن حروفه ستصبح **Bold**، ويتحول الملف إلى **read only** بعد عمل **compilation** للتطبيق. مع العلم أن كل تطبيق يوجد فيه ملف رئيسي واحد فقط.




**الشكل (٩ - ١) نموذج لتحديد الملف الرئيسي في project manager**

## ب . تحميل بيئة التطبيق (Initializing the Environment)

المهمة الأولى التي يجب أن يتضمنها عمل الملف الرئيسي هو تحديد وتحميل بيئة التطبيق. البيئة البديهية للتطبيق VFP يولد مجموعة معينة من إيعازات SET و متغيرات النظام عند تحميل VFP. على أية حال فإن هذه التحديدات قد تكون غير ملائمة للتطبيق الخاص بنا ويجب أن نغيرها . ولغرض معرفة البيئة الحالية للتطبيق نتبع ما يلي:-

من قائمة Tool نختار Options.

نضغط مفتاح Shift مع الضغط على اختيار OK، فيقوم VFP بعرض بيئة SET في Command Window.

في بيئة التطبيق يمكن تحديد مواصفات التطبيق ودرج الفعاليات التالية فيه:-

تحديد المتغيرات.

تحديد المسار البديهي (Default Path).

فتح قواعد البيانات الخاصة بالتطبيق.

تحديد المكتبات الخارجية وملفات الإجراءات.

ملاحظة:- قد تكون واحدة من الأفكار الجيدة هي في خزن التحديدات البديهية التابعة إلى VFP في برنامج أو متغيرات من نوع Public وذلك لغرض استعادتها بعد الانتهاء من التطبيق الخاص بنا والرجوع إلى VFP.

**ج . عرض واجهة التطبيق الرئيسية**

قد تكون واجهة المستخدم الرئيسية هي عبارة عن قائمة أو نموذج أو أي عنصر مرئي آخر، حتى ولو كان التطبيق يعرض شاشة دخول إلى النظام (logon screen) قبل عرض واجهة الاستخدام الرئيسية. يمكن عرض الواجهة الرئيسية عن طريق كتابة الإيعاز do في برنامج main إذا كانت الواجهة عبارة عن برنامج، أو do form إذا كانت الواجهة عبارة عن نموذج.

**د . السيطرة على دورة الأحداث**

بعد تحديد البيئة وعرض واجهة التطبيق الرئيسية، يجب التهيؤ لتحديد دورة الأحداث في انتظار تدخل المستخدم (user interaction). ويتم السيطرة على دورة الأحداث من خلال الإيعاز READ EVENTS، والتي تسبب في بدء VFP في معالجة الأحداث من قبل المستخدم (مثل عمل click، أو ضغط مفتاح في لوحة المفاتيح).
من المهم جدا وضع إيعاز READ EVENTS في الموقع الصحيح من الملف الرئيسي، وذلك لأن كل المعالجات تتوقف من لحظة كتابة الإيعاز لحين الوصول إلى الإيعاز المعاكس والذي هو CLEAR EVENTS. ولعل أفضل موقع له هو في نهاية البرنامج الرئيسي، مع ملاحظة أنه إذا لم نكتب هذا الإيعاز فإن التطبيق سوف يعود مرة ثانية إلى نظام التشغيل بعد تنفيذه مباشرة. وبعد البدء في دورة الأحداث فإن التطبيق سوف يكون تحت سيطرة الواجهة الرئيسية للتطبيق. والمثال التالي يوضح كتابة هذا الإيعاز في الملف الرئيسي للتطبيق:-

```
Do Form Startup
Read Events
```

وبالمقابل إذا أردنا أن نقطع دورة الأحداث فإننا نستخدم إيعاز CLEAR EVENTS من أحد اختيارات Menu أو كاستجابة لحدث click على Button في النموذج الرئيسي. مع التأكيد على وجود إيعاز قطع دورة الأحداث قبل كتابة إيعاز بدء دورة الأحداث لمنع التطبيق من الدخول في دورة غير منتهية (infinite loop).

**هـ. استرجاع البيئة الأساسية**

من الأمور المهمة هي استعادة البيئة الأساسية لتطبيق VFP بعد الانتهاء من التطبيق الخاص بنا. ويمكن استعادة هذه البيئة إذا كان هناك خزن مسبق لقيم SET سواء في متغيرات من نوع Public أو في برنامج فرعي محدد فإننا سنقوم بتحميل هذا البرنامج أو استعادة قيم المتغيرات قبل الخروج من التطبيق ويفضل ان يتم استدعاءها من برنامج رئيسي . يبين الجدول التالي أهم الأجزاء التي يمكن أن يتضمنها البرنامج الرئيسي:-

| Code | Comments |
|---|---|
| DO SETUP.PRG | Call program to set up environment (store values in public variables) |
| DO MAINMENU.MPR<br>Or<br>DO FORM STARTUP.SCX | Display menu as initial interface<br>Or<br>Display form as initial interface |
| READ EVENTS | Establish event loop. A different program (such as Mainmenu.mpr must issue a CLEAR EVENTS command( |
| DO CLEANUP.PRG | Restore environment before quitting |

**و . بناء التطبيق من مشروع**

الخطوة الأخيرة في عمل compilation لمشروع هو بناء التطبيق الخاص به. والنتيجة النهائية لهذه العملية هي عبارة عن ملف واحد يحتوي على جميع ملفات المشروع، ويمكن نقل قاعدة البيانات والجداول مع التطبيق لغرض استخدامها من قبل المستفيد . ولغرض فحص المشروع، نختار build من project manager ومن ثم نختار rebuild project من شاشة build options التي ستظهر لاحقا.

ولغرض بناء التطبيق من المشروع نختار Build من project manager، ومن ثم نختار Build Application من شاشة build option. فيتولد ملف التطبيق من نوع (.app) والذي يمكن نقله (مع البيانات) إلى حاسبات المستخدمين (بشرط وجود تطبيق VFP على هذه الحاسبات).

ولغرض بناء تطبيق تنفيذي (executable) نختار Build من project manage، ومن ثم نختار Build Executable من شاشة build options. فيتولد ملف التطبيق التنفيذي من نوع (.exe) والذي يمكن نقله (مع البيانات) إلى حاسبات المستخدمين (ولا يشترط في هذه الحالة وجود تطبيق VFP على هذه الحاسبات).

تنبيه:- في حالة نقل تطبيق تنفيذي (.exe) إلى حاسبة لا تحتوي على التطبيق VFP فيجب نقل ملفات معينة موجودة في الحاسبة التي تم بناء التطبيق عليها وبالتحديد في المسار التالي:-

\..windows\system

وهذه الملفات من نوع Dynamic Link Libraray (.dll) وهي الملفات التالية:-

**vbame.dll**
**vfp6r.dll**
**vfp6renu.dll**

**ولضمان عد حصول أي خطأ في التنفيذ يفضل نقل الملف vfp6run.exe الموجود في نفس المسار أعلاه.**

## ١٠ . تقنيات تطويرية للاستخدام المتعدد Multi-User Development Techniques

بالنسبة لكل مطوري برامج VFP يجب تبني مفهوم معالجة الاستخدام المتعدد والذي ينشأ من خلال تعامل عدد من المستفيدين عبر شبكة حاسبات مع البيانات الموجودة في قاعدة بيانات معينة. والتعبير الأكاديمي لهذه العملية هو ***Concurrency Control*** وذلك لأن المشكلة تحدث كنتيجة لمعالجة متزامنة للبيانات. والتداخل الذي يحدث هو أن مستخدمين أو أكثر يحاولون في نفس الوقت معالجة نفس البيانات. يوفر VFP تقنيات جيدة لمعالجة تداخل الاستخدام المتعدد، وكذلك يوفر إستراتيجيات معالجة جداول منفردة (Buffering modes) وإستراتيجيات معالجة جداول متعددة (Transaction processing).

**أ . مفاهيم الاستخدام المتعدد (Concepts of Multi-User)**

إذا كان لدينا شبكة محلية LAN أو شبكات Intranet و Web، ففي بعض الأحيان قد يدخل أكثر من مستفيد إلى البيانات لغرض تحديثها، وقد يكون من محاسن الصدف عدم حدوث هذا التداخل، إلا أن احتمالية حدوث هذا التداخل تتزايد بازدياد عدد المستخدمين للنظام، إذا لم نرغب أن نترك الأمر لمشيئة الصدف فيجب علينا توفير المعالجة المناسبة لمثل هذه الحالات من التداخل.

معالجة تداخل الاستخدام المتعدد في VFP يتم عن طريق استخدام عدد من إيعازات SET وعدد من إيعازات LOCK. عملية القفل (Locking) توفر للمستخدم عملية منع المستخدمين الآخرين من تحديث البيانات التي يتم معالجتها من قبل المستخدم الذي يقوم بعملية Locking. وقد يكون من الأمور المريحة بالنسبة لمطوري برامج VFP من أن هذه الحلول لمعالجة الاستخدام المتعدد تحدث بصورة أوتوماتيكية من قبل التطبيق بدون الحاجة لكتابة الإيعازات لحل هذه التداخلات. في تطبيق VFP تتوفر بعض تصرفات القفل البديهية ( Default Locking Behavior) والتي تتطلب عبارات برمجية قليلة جدا، ويقوم VFP بمعالجة حالات التداخل وإرسال الرسائل (Messages) إلى المستخدمين. ولكن لا وجود لاستراتيجية موحدة لمعالجة كل حالات التداخل التي تحدث، لذلك يجب الإلمام بكل الإيعازات الخاصة بهذا المجال والاستراتيجيات المتوفرة في VFP.

**ب . حالات حدوث التداخل في الاستخدام المتعدد**

هناك على الأقل ثلاث حالات من حالات التداخل بين المستخدمين وهي:-

**اولا . Edit Conflicts (تداخل التحديث):-** وهي الحالة التي تحدث عند محاولة أكثر من مستخدم من الدخول على البيانات لغرض التحديث، السؤال هنا هل نريد أن يقوم كل المستخدمين بعملية تحديث البيانات في نفس الوقت أو نسمح لمستخدم واحد من تحديث البيانات تاركا المستخدمين الآخرين في انتظار دورهم بعد انتهاء هذا المستخدم من عملية التحديث.

**ثانيا . Query Conflicts (تداخل الاستفسار):-** هذه الحالة تحدث عندما ينفذ أحد المستخدمين استفسارا أو تقريرا يأخذ فترة زمنية طويلة نسبيا، وقد يكون نوع الاستفسار أو التقرير هو من الأنواع التي تتطلب ثبات

البيانات في الجداول المتعلقة، فإذا سمحنا لبقية المستخدمين من معالجة الجداول الداخلة في الاستفسار أو التقرير فإنه يمكن حدوث تغيير في قيم بعض البيانات مسببة حدوث أخطاء في النتائج.

**ثالثا . Maintenance Conflicts (تداخل الصيانة):-** تحدث هذه الحالة عند قيام أحد المستخدمين بعمل إجراءات الصيانة للبيانات والتي قد تتضمن إعادة فهرسة البيانات أو عمل Backup للبيانات أو حذف قيود من جداول البيانات (Pack). مما يتطلب التعامل مع البيانات بأسلوب منع بقية المستخدمين من التعامل مع البيانات في نفس الوقت.

**تنبيه:-** عملية التداخل لا يمكن تجنبها عن طريق إهمال احتمالية حدوثها، وذلك لأنه حتى بالنسبة لحاسبة منفردة غير مربوطة على شبكة حاسبات (Single User)، فإن إمكانيات أنظمة التشغيل متعددة المهام ( Multi-Tasks Operating systems) وبضمنها Windows، ومع وجود واجهات المستخدم الرسومية ( Graphical User Interface) فإن المستخدم يستطيع تنفيذ التطبيق أكثر من مرة، أو تنفيذ عدة نماذج في نفس الوقت وقد تتعامل هذه النماذج مع نفس البيانات.

## ج . إستراتيجيات القفل في VFP

إذا قررنا أنه في وقت واحد يوجد فقط مستخدم واحد يمكنه تحديث البيانات فإننا نستخدم Lock. يقوم VFP وبالتعاون مع نظام التشغيل من الاستجابة للقفل وفتح القفل على البيانات من التطبيق الذي يتم تنفيذه من قبل عدد من المستخدمين.

يتم استخدام Locking لضمان التحديث الثابت للبيانات في القاعدة، وقد يكون القفل للجدول بأكمله (مثلا في حالة تغيير الهيكل (modifying structure)) أو يكون القفل على قيد (أو قيود) محدد ضمن الجدول. توفر ماكينة VFP (VFP Database Engine) ضمان عدم تحديث البيانات من قبل أكثر من مستفيد في نفس الوقت حتى ولو لم يتم استخدام Locking.

## د . الأنواع الرئيسية لقفل البيانات

من الأمور المتعارف عليها في أنظمة إدارة قواعد البيانات وجود ثلاثة أنواع من Locking. ويبين الجدول أدناه هذه الأنواع الثلاثة.

| ت | نوع القفل Locking Type | متوفرة في VFP | مستخدم القفل | | بقية المستخدمين | |
|---|---|---|---|---|---|---|
| | | | قراءة | كتابة | قراءة | كتابة |

| | | | | | | |
|---|---|---|---|---|---|---|
| ١ | Exclusive | نعم | نعم | نعم | كلا | كلا |
| ٢ | Shared | كلا | نعم | كلا | نعم | كلا |
| ٣ | Update | نعم | نعم | نعم | نعم | كلا |

**جدول يوضح أنواع Locking في أنظمة إدارة قواعد البيانات**

## هـــ .قفل قاعدة البيانات Database Locking

عند فتح قاعدة البيانات في VFP فإننا نفتحها بطريقتين فقط، إما نفتحها للاستخدام المتعدد أو نفتحها للاستخدام المفرد. ويعتبر هذا المستوى من القفل أعلى المستويات ولا يتم التفكير مطلقا في الاستخدام المتعدد إذا تم فتح قاعدة البيانات للاستخدام المفرد لأن هذا الأمر لا يمكن تحقيقه وقاعدة البيانات مفتوحة للاستخدام المفرد. وأي محاولة لتنفيذ التطبيق المفتوح للاستخدام المفرد أو استخدام إيعازات القفل تكون استجابة VFP من خلال رسالة (Message) تظهر فيها عادة عبارة (File Access Denied).

ويمكن فتح قاعدة البيانات من خلال الإيعازات التالية:-

لفتح قاعدة البيانات للاستخدام المنفرد:-

open database [اسم قاعدة البيانات] exclusive

لفتح قاعدة البيانات للاستخدام المتعدد:-

open database [اسم قاعدة البيانات] shared

لفتح قاعدة البيانات مع أخذ نوع الفتح من بيئة التطبيق أو من خلال إيعاز Set Exclusive:-

open data[اسم قاعدة البيانات]

فإذا كان الإيعاز المستخدم في الملف الرئيسي (set exclusive on) أو كانت البيئة هي Exclusive on فإن القاعدة تفتح للاستخدام المنفرد، أما إذا كان الإيعاز المستخدم في الملف الرئيسي (set exclusive off) أو كانت البيئة هي Exclusive Off فإن القاعدة تفتح للاستخدام المتعدد.

## و . قفل الجداول Table Locking

إيعاز قفل الجدول يشبه إيعاز قفل قاعدة البيانات، فإذا أردنا فتح جدول للاستخدام المنفرد نستخدم الإيعاز التالي:-

use [اسم الجدول] exclusive

وفي هذه الحالة فإن بقية المستخدمين لا يستطيعون القراءة أو الكتابة إلى هذا الجدول.

وإذا أردنا أن نفتح جدول للاستخدام المتعدد نستخدم الإيعاز التالي:-

use [اسم الجدول] shared

## ز . قفل الجداول المفتوحة للاستخدام المتعدد

حتى إذا فتحنا الجدول للاستخدام المتعدد فإننا نستطيع عمل Update Lock عليه من خلال استخدام الدالة Flock()، أي أننا نسمح لمستخدم واحد (الذي يعمل Lock) من استخدام الجدول للتحديث وبقية المستخدمين يمكنهم القراءة من الجدول فقط. الدالة Flock() ترجع قيمة .T. في حالة نجاح القفل، وترجع .F. في حالة كون الجدول مقفولا من قبل مستخدم آخر. ونستطيع فتح القفل من خلال إيعاز unlock، أو إذا عملنا unlock لأي قيد في الجدول المقفول. وهناك دالة أخرى يقترن عملها بهذا المجال وهي دالة Isflocked([table name]) والتي ترجع .T. إذا كان الجدول مقفولا ويرجع .F. إذا كان الجدول غير مقفول.

تنبيه:- إذا أردنا فتح الجدول لعمل فعاليات الإدامة عليه (re-indexing، structure modifying، Packing) فيجب فتح الجدول exclusively لمنع بقية المستخدمين من القراءة من الملف أو الكتابة فيه.

## ح . قفل القيود المستقل Explicit Row Buffering

لا يمكن عمل exclusive لقيد ضمن جدول، إلا أنه يمكن فقط عمل Lock على القيد من خلال استخدام الدالة Lock() والدالة Rlock() والتي ترجع .T. في حالة كون القيد مقفولا وترجع .F. في حالة كون القيد غير مقفول. والمثال التالي يوضح استخدام القفل:-

```
SELECT 0
USE customer
Go top
IF !RLOCK()
     LOCK()
ENDIF
UNLOCK IN customer
```

كما ويوفر VFP إمكانية عمل قفل على عدد من القيود عن طريق استخدام الإيعاز:-

```
Set multilocks on
```

وتوجد دالة أخرى يقترن عملها بهذا الموضوع وهي دالة Isrlocked() والتي ترجع .T. في حالة كون القيد مقفولا وترجع .F. في حالة كونه غير مقفول.

## ط . قفل القيود الضمني Implicit Row Buffering

يقوم VFP بعمل update lock للقيود بصورة أوتوماتيكية دون تدخل مطور التطبيق، ويقوم بضمان عدم حدوث وجود أكثر من تحديث في نفس الوقت لبيانات مشتركة بين أكثر من مستخدم، ولفهم هذا القفل نأخذ مثالا في وجود مستخدمين تم دخولهم إلى نفس الجدول وهما يقفان على نفس القيد والجدول مفتوح للاستخدام المتعدد، فيمكن لهما تحديث القيد في نفس الوقت، إلا أنه في الحقيقة يقوم VFP من إجراء تحديث واحد فقط في نفس الوقت تماما، لذلك فإن قيمة الحقول المحدثة لهذا القيد ستكون القيم التي أدخلها المستفيد الذي تم قبول تحديثه كآخر التحديثات (التحديث من نصيب من يحدث أخيرا).

**ي . كيفية الاستجابة في حالة عدم الحصول على LOCK**

في حالة محاولة عمل LOCK على جدول أو قيود وفشل المحاولة لكون أن هناك مستخدم آخر عمل LOCK، فالذي يحدد عدد محاولات عمل LOCK هو إيعاز set reprocess حيث يتم تحديد عدد مرات المحاولة أو وقت المحاولة قبل ظهور عبارة الخطأ. والصيغة العامة لهذا الإيعاز هو:-

```
Set Reprocess to [option]
```

وعبارة option ممكن أن تكون أحد العبارات التالية:-

عدد مرات تكرار محاولات القفل.

وقت التكرار بالثواني.

**ك . حجز القيد وحجز الجدول Row And Table Buffering**

بالإضافة إلى ما تم ذكره من إيعازات (وهي إيعازات موجودة في الإصدارات السابقة أيضا) فإن VFP يوفر إستراتيجيات جديدة وهي:-

Row and Table buffering.

Transaction Processing.

الفرق بين الإستراتيجيتين هي أن row and table buffering يخص الجداول المنفردة و transaction process يخص عدد من الجداول ضمن قاعدة البيانات.

هذه الإستراتيجيات غير متضاربة فيما بينها، حيث توفر الأولى طريقة لحذف تحديثات المستخدم في حالة الجداول المنفردة سواء كانت داخل القاعدة أو بشكل جداول حرة، والأخرى تضمن أن التحديثات لأكثر من جدول إما أن تقبل للكل وإما أن ترفض للكل بشرط أن تكون كل الجداول ضمن قاعدة البيانات.

ويمكن النظر إلى الإستراتيجيتين على أنهما تحجز البيانات لأنها تقوم بإنشاء قيم temporary للبيانات، ففي حالة update تجري التعديل للجدول من خلال هذه القيم وفي حالة revert تهمل هذه القيم وتبقى بيانات جدول مثل ما كانت.

والجدول التالي يوضح مقارنة بين الإستراتيجيتين:-

| | Single free table | Single DB table | Multiple free tables | Multiple DB tables |
|---|---|---|---|---|
| Row &Table | Yes | Yes | No | No |
| Transaction | No | Yes | No | Yes |

**جدول يبين الفرق بين استراتيجيات الحجز**

**ل . الحجز المتسامح والمتشدد Optimistic and Pessimistic Locking**

عند التوجه إلى استخدام الحجز، فإن القرار الأساسي الذي يجب اتخاذه هو في استخدام الحجز المتسامح (optimistic buffering) أو الحجز المتشدد (pessimistic buffering) والقرار الثاني هو في حجز الجدول بأكمله أم في حجز قيد واحد فقط من الجدول. ماكينة قاعدة بيانات VFP ( Visual Foxpro Database Engine) يوفر نوعي الحجز، ومن الجدير بالذكر هنا أن الحجز هي ليست خاصية (property) تابعة إلى الجدول ضمن قاعدة البيانات، وإنما هي خاصية تابعة إلى الجدول عندما يكون مفتوحا ضمن Data Environment داخل Form. ولذلك فإن الجدول يمكن حجزه في Form بأكمله وفي Form آخر نحجز فيه قيد واحد فقط، أو قد نحجز الجدول حجز متسامح في Form وفي Form آخر نحجزه حجز متشدد.

## اولا . أساسيات القفل المتسامح Essentials of Optimistic Locking

في الحجز المتسامح (optimistic) يتم تأجيل القفل (lock) لحين وقت إيعاز الموافقة على التحديث، لذلك يمكن أن يقوم أكثر من مستخدم من تحديث نفس القيد في نفس الوقت.

عندما يكون لدينا optimistic row buffering فإن VFP سيقوم بتحديث هذا القيد عند حركة مؤشر القيود (record pointer)، أو عند غلق الجدول، أو عند توليد إيعاز tableupdate()، ففي هذه اللحظة فقط إذا حاول مستخدم آخر تحديث هذا القيد فسيولد VFP خطأ.

وعندما يكون لدينا optimistic table buffering فإن التحديثات على البيانات سوف تؤجل إلى أن يقوم المستخدم بغلق الجدول أو استخدام إيعاز tableupdate()، وعندها يقوم VFP بعمل اختبار لكل قيد في الجدول، فإذا قام مستخدم آخر بتحديث ولو قيد واحد بعد بدء عملية الاختبار فإن VFP سيولد خطأ.

يعتبر optimistic row buffering ذو فائدة كبيرة عندما يقوم المستخدم بتحديث الجدول بشكل قيد واحد في كل مرة، كما ويتيح لأكثر من مستخدم من تحديث نفس القيد، وأيضا فإن هذه الطريقة مفيدة لتقليل وقت القفل ولزيادة الكفاءة كما و تتيح هذه الطريقة لأكثر من مستخدم من تحديث مجموعة قيود في جدول معين.

## ثانيا . أساسيات القفل المتشدد Essential of Pessimistic Locking

في pessimistic locking فقط يوجد مستخدم واحد يقوم بتحديث القيد في نفس الوقت، حيث يقوم VFP بعمل row-level update أو table-level update خلال فترة التحديث. وعند استخدام pessimistic row locking فسوف يقوم VFP وبصورة أوتوماتيكية بقفل القيد لحين حركة record pointer أو استخدام إيعاز tableupdate() أو إغلاق الجدول.

كذلك فإن VFP يقوم بقفل كل القيود التي يقوم المستخدم بتحديثها لحين غلق الجدول أو استخدام tableupdate()، ولا يرى بقية المستخدمين التحديثات التي يقوم بها المستخدم لحين الانتهاء.

تكون pessimistic row buffering مفيدة عندما يقوم المستخدم بتحديث القيود في جدول بشكل قيد في كل مرة، وتكون البيانات حساسة جدا بحيث لا يسمح إلا لمستخدم واحد من تغييرها في نفس الوقت، ويمكن استخدام هذه الطريقة عندما يكون طول وقت القفل غير مهم أو يسمح النظام للمستخدم من قفل القيد لوقت غير محدد.

تكون pessimistic table buffering مفيدة بنفس خصائص وفوائد pessimistic row buffering، إضافة إلى أن المستخدم يستطيع قفل أكثر من قيد في نفس الوقت أو قفل الجدول بأكمله.

**م . الأحداث المتعارف عليها في تطبيقات الاستخدام المتعدد**

هناك ثلاثة أحداث رئيسية في الاستخدام المتعدد قد تسبب تداخل بين المستخدمين وهذه الأحداث هي:- إضافة قيد (ADD)، تحديث قيد (UPDATE)، حذف قيد (DELETE)، وفيما يلي شرح لكل حالة عند استخدام BUFFERING:-

**اولا . ADD:-** في حالة إضافة قيد جديد فإن التداخل لن يحدث أبدا إذا كان Buffering في حالة Enabled وذلك لأن القيد الجديد يتم حجزه ولا يظهر للمستخدمين الآخرين إلا إذا تم قبوله في الجدول.

**ثانيا . UPDATE:-** في حالة استخدام pessimistic فإن التداخل لن يحدث أبدا، ولكن عند استخدام optimistic فقد يحدث تداخل عندما يحاول مستخدم معين تحديث وقبول تحديث قيد بينما هناك مستخدم آخر يقوم بتحديث نفس القيد، ففي هذه الحالة يسمح VFP في احتواء التداخل بالنسبة لتطبيق المستخدم الثاني وذلك من خلال شاشة استفسار حول رغبة المستخدم في الكتابة على تحديث المستخدم الآخر أو قبول تحديث ذلك المستخدم.

**ثالثا . DELETE:-** هنا قد يحدث التداخل بطريقتين، الطريقة الأولى عندما يحاول مستخدمين حذف نفس القيد في نفس اللحظة، ففي هذه الحالة لا توجد مشكلة لأن كليهما يقومان بنفس العمل. والطريقة الثانية هي عندما يحاول مستخدم حذف قيد هو تحت تحديث مستخدم آخر. الأسلوب الأفضل لحل هذا التداخل هو جعل فعالية الحذف (أو delete button) يقوم بتحسس فيما إذا كان القيد المراد حذفه هو تحت تحديث مستخدم آخر ويقوم برفض عملية الحذف.

**ن . Transaction Process**

يعرف Database Transaction على أنه وحـــدة عمـــل منطقيـــة ( a logical unit of work). يوفر VFP عملية Transaction Process للسماح بتغطية كل تحديثات قاعدة البيانـــات معـــا ومعالجتها كوحدة عمل واحدة. بعدها يمكن قبول أو رفض كل التحديثات.

VFP يسمح للمستخدم في تعريف transaction من خلال إيعـــاز Begin Transaction. وعنـــد تطبيق هذا الإيعاز فإن كل التحديثات على جداول قاعدة البيانات سوف تكتب في مـــا يـــسمى transaction buffer، ولا تحدث البيانات في الجداول مباشرة. ولغرض إلغاء التحديثات أو رفـــضها يمكـــن اســـتخدام إيعـــاز Rollback، لتعود البيانات كما كانت (وذلك لأن البيانات موجودة في transaction buffer. وفـــي حالة حدوث توقف في الحاسبة أو إطفائها مباشرة فإن عملية rollback تحدث بـــصورة أوتوماتيكيـــة. ولقبـــول التحديثات نستخدم إيعاز End Transaction. ومن الجدير بالذكر أن transaction process لا يطبق إلا على الجداول الموجودة في قاعدة البيانات فقط ولا يمكن أن تطبق على الجداول الحرة.

ويبين الجدول التالي أنواع buffering في VFP

| **To enable** | **Use this value** |
|---|---|
| No buffering .The default value. | ١ |
| Pessimistic record locks which lock record now, update when pointer moves or upon TABLEUPDATE( ). | ٢ |
| Optimistic record locks which wait until pointer moves, and then lock and update. | ٣ |
| Pessimistic table locks which lock record now, update later upon TABLEUPDATE( ). | ٤ |
| Optimistic table lock which wait until TABLEUPDATE( ), and then lock and update edited records. | 5 |

**تنبيه:-** يجب عمل multilock في حالة on عند استخدام أنواع buffering من ٢ فما فوق.

## ١١ . الصنوف Classes

تعتبر الصنوف من أهم الأجزاء في لغات البرمجة بالكائنات الموجهة ، فهي تعتبر القوالب الأساسية التي تصاغ منها الكائنات في التطبيق. ويمكن النظر إلى مفهوم الصنوف من وجهة نظر أخرى ألا وهي اعتبارها نوع من أنواع البيانات غير التقليدية والمعقدة (وذلك لاحتوائها على العبارات البرمجية والبيانات و الإجراءات مع بعضها) فمن الممكن استخدامها مثل استخدام أنواع البيانات في أية لغة برمجية تقليدية، ففي البرمجة التقليدية يمكن تعريف متغير معين على أنه متغير حرفي أو متغير رقمي...الخ. وفي مفهوم التعامل مع الصنف كنوع بيانات يمكن تعريف أي كائن (مثل المتغير) ليكون من نفس الصنف الذي تم تعريفه تحته.

يمكن استخدام الصنوف بطرق مختلفة. التخطيط الذكي يتيح للمستخدم معرفة الصنف المناسب لكل مهمة في التطبيق. ويمكن للمستخدم إنشاء صنف لكل control أو لكل form يمكن ان يستخدمه، ولكن هذه الطريقة ليست هي الأفضل في بناء التطبيقات. حيث ستنتهي هذه الطريقة إلى مجموعة من الصنوف التي تؤدي نفس المهام تقريبا إلا أنه يجب إدامة وتحديث أو تغيير خصائص كل صنف على حدة. لذلك يمكن إنشاء صنوف لأداء مهام عامة، فمثلا يمكن إنشاء button command يتيح للمستخدم الانتقال عبر قيود جدول معين ويمكن إضافة هذا button إلى أي form لأداء هذه المهمة.

كما ويمكن إنشاء forms أو controls ذات مواصفات وخصائص محددة واستخدامها في التطبيق حتى تظهر جميع عناصر التطبيق بصورة موحدة. فمثلا يمكن تحديد graphics أو ألوان معينة للنماذج أو المسيطرات التي يتم استخدامها في النماذج، أو مثلا إنشاء textbox ذات مواصفات محددة من حيث نوع الحرف وحجمه واتجاه الكتابة والألوان واستخدامه في أي نموذج داخل التطبيق.

## أ . الصنوف الأساسية في VFP (Visual FoxPro Base Classes)

يبين الجدول التالي الصنوف الرئيسية في VFP والتي يمكن بناء صنوف ثانوية عن طريق هذه الصنوف الرئيسية وإضافتها إلى مكتبات الصنوف واستخدامها في التطبيقات.

| | | | |
|---|---|---|---|
| ActiveDoc | Custom | Label | PageFrame |
| CheckBox | EditBox | Line | ProjectHook |
| Column* | Form | ListBox | Separator |
| CommandButton | FormSet | OLEBoundControl | Shape |
| CommandGroup | Grid | OLEContainerControl | Spinner |
| ComboBox | Header* | OptionButton* | TextBox |
| Container | Hyperlink Object | OptionGroup | Timer |
| Control | Image | Page* | ToolBar |

**جدول الصنوف الرئيسية في VFP**

ملاحظة:- الصنوف المؤشرة بعلامة (*) في الجدول أعلاه يعني أن هذا الصنف هو مكمل للصنف الأب ولا يمكن إنشاء صنف ثانوي منه.

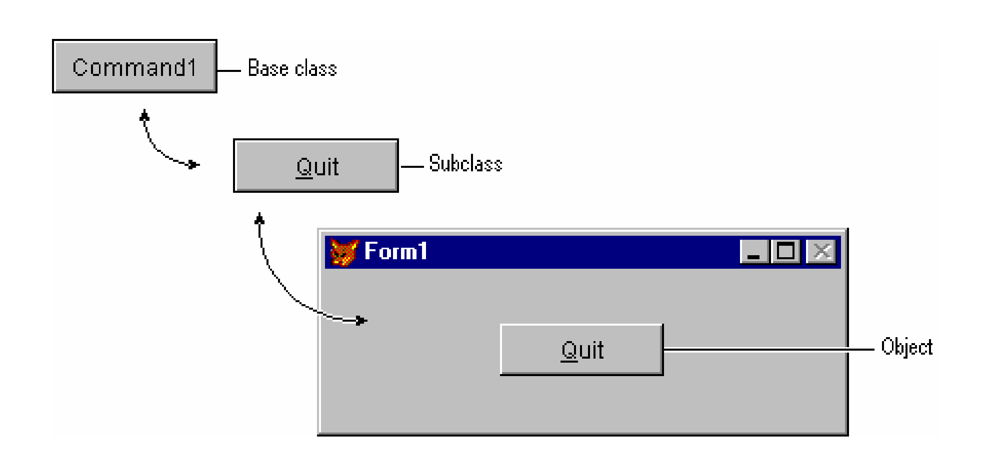
## ب . التوسع بالصنوف الرئيسية

يمكن إنشاء صنوف ثانوية من الصنوف الرئيسية لوضع خصائص بديهية لها. فمثلا إذا أردنا أن نجعل الاسم لمسيطر معين هو اسم نستخدمه في إنشاء النماذج الخاصة بنا فيمكن عمل ذلك من خلال إنشاء صنف جديد مبني على control المطلوب بالاسم الذي نريده. وكذلك يمكن إنشاء صنف خاص إلى forms من حيث المظهر والاتجاه والألوان ويكون هو القالب الذي يتم بناء جميع forms التطبيق عليه. كما ويمكن تحديد عمل المسيطر في الصنف لأداء عمل معين، فمثلا إذا أردنا أن نعمل command button يستجيب لحدث click عليه بأن يغلق النموذج الذي يوضع فيه، فيمكن عمل ذلك عن طريق إنشاء صنف ثانوي مبني من الصنف الأب (command button) مع كتابة الإيعاز التالي فيه في الحدث click:-

ThisForm.Release

ويمكن توليد كائنات تابعة لهذا الصنف الجديد في أي نموذج، وسوف يؤدي هذا الكائن نفس العمل أينما وضع (وهي عملية إغلاق النموذج). ويوضح الشكل (١١-١) هذا المثال.

عند إنشاء الصنوف الثانوية فليس بالضرورة أن يحتوي الصنف الجديد على مسيطر واحد ومن نوع واحد فقط، حيث يمكن إضافة عدد من المسيطرات إلى الصنف الجديد وبأعداد مختلفة وحسب حاجة التطبيق. فمثلا إذا أردنا أن نعمل صنف جديد يحتوي على العمليات التي يمكن أن تجري على جدول في نموذج معين فإننا سنضيف مثلا عددا من command buttons يساوي عدد العمليات التي يمكن أن تحدث على الجدول.




**الشكل (١١-١) مثال لإنشاء صنف ثانوي من صنف رئيسي**

## ج . إنشاء صنوف غير مرئية Creating Non-Visual Classes

تعرف الصنوف غير المرئية على أنها الصنوف التي لا يوجد لها عنصر مرئي أثناء وقت التنفيذ (run time)، ويمكن أن تكون مرئية فقط أثناء وقت التصميم (design time). فمثلا يمكن إنشاء صنف جديد وليكن اسمه StrMethods يحتوي على عدد من الصيغ (methods) الخاصة بالتعامل مع السلاسل الحرفية (strings)، فيمكن مثلا إضافة هذا الصنف إلى form يحتوي على editbox ويمكن استدعاء الصيغ في هذا الصنف في أي وقت نشاء، فعلى سبيل المثال من الممكن أن يحتوي هذا الصنف على صيغة باسم WordCount يقوم بحساب طول السلسلة الحرفية في editbox وليكن اسمه edit1، فيمكن استدعائه بالصيغة التالية:-

store ThisForm.StrMethods.WordCount(ThisForm.edit1.Value) to s1

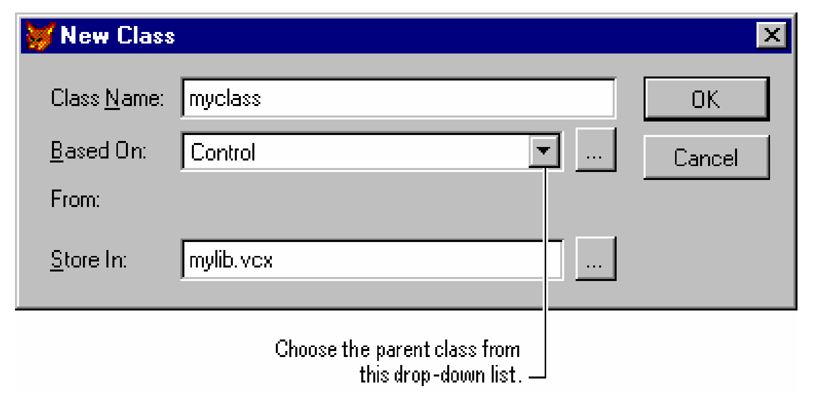
وكما ذكرنا سابقا فإن Non-Visual Classes لها جزء مرئي في وقت التصميم فقط ولا يوجد لها جزء مرئي في وقت التنفيذ، ويمكن تغيير خاصية picture لهذا الصنف إلى ملف من نوع (bmp.) ليكون معروفا في وقت التنفيذ ويمكن تمييزه عن بقية الصنوف.

**د . إنشاء الصنوف**

يمكن إنشاء الصنوف من خلال استخدام مصمم الصنوف (class designer) وكما يلي:-

من قائمة file نختار new ومن ثم نختار class.

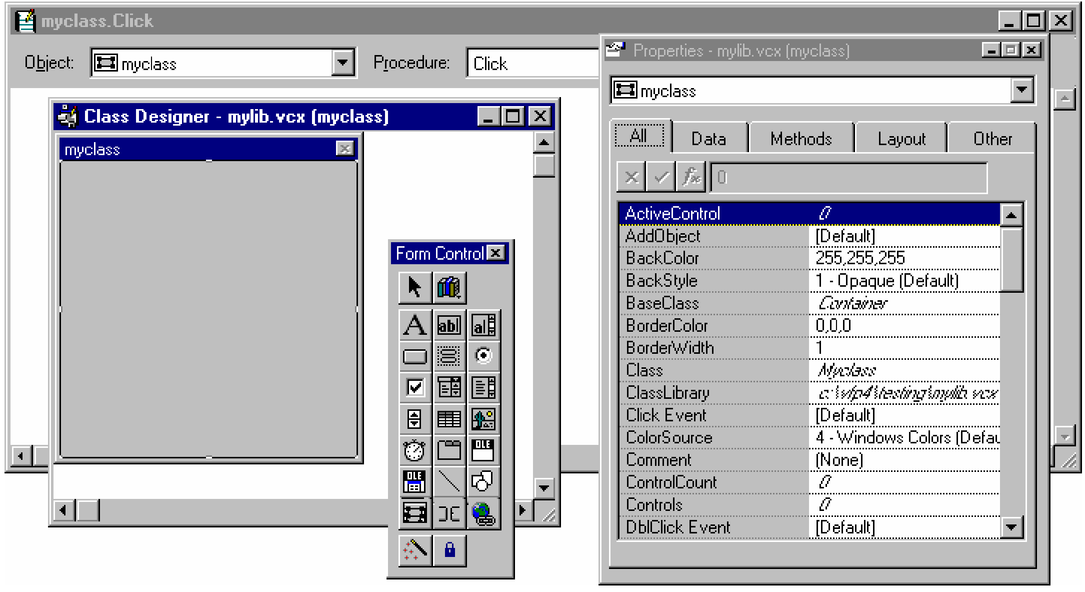
تظهر لنا شاشة إنشاء صنف جديد، كما في الشكل (١١-٢).




**الشكل (١١-٢) شاشة إنشاء صنف جديد**

حيث يمثل مربع Class Name اسم الصنف الذي نرغب في تسميته به. ويمثل مربع Based On الصنف الرئيسي الذي نريد أن ننشأ الصنف الثانوي منه. ويمثل مربع Store In اسم المكتبة التي نريد خزن الصنف الجديد فيه.

وبعد تعريف اسم الصنف ومكتبة الخزن والصنف الأب واختيار OK ستظهر لنا شاشة مصمم الصنوف ( Class Designer) كما موضح في الشكل (١١-٣).




**الشكل (١١-٣) شاشة مصمم الصنوف (Class Designer)**

يوفر مصمم الصنوف نفس واجهة مصمم النماذج، حيث يسمح للمستخدم من تحديث الخصائص من خلال نافذة الخصائص (Properties window)، كما ويسمح بكتابة الإيعازات للأحداث داخل نافذة التحديث ( editing window).

**هـ . إضافة كائنات إلى Class Control أو Container Class**

إذا تم إنشاء الصنف الجديد من خلال صنف الأب الذي قد يكون control أو container، فيمكن إضافة controls له بنفس الطريقة كما في مصمم النماذج. وبغض النظر عن الصنف الأب فإنه يمكن تحديث الخصائص وكتابة الإيعازات للصيغ في هذا الصنف الجديد.

كما ويمكن إضافة خصائص وصيغ جديدة للصنف الذي يتم إنشاؤه مثل ما نرغب. حيث إن الخصائص تحمل قيم معينة، والصيغ يحمل إيعازات برمجية محددة يتم تنفيذها حين استدعاء الصيغة.

ولإضافة خاصية جديدة للصنف نتبع ما يلي:-

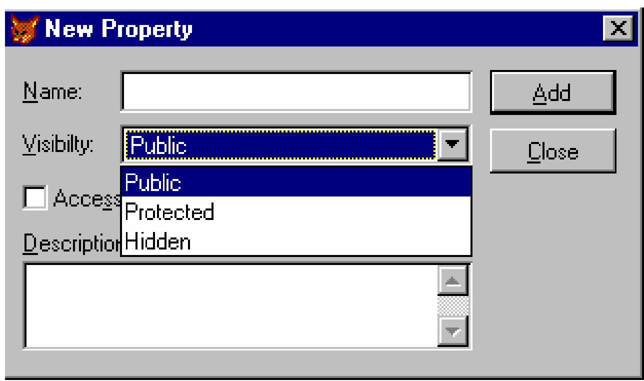
من قائمة Class نختار New Property.

في مربع New Property نطبع اسم الخاصية.

نحدد النوع من خلال اختيارات:-Public، Protected، أو Hidden.

إذا كان النوع Public، فإن الخاصية يمكن تغييرها في أي مكان في التطبيق، أما بالنسبة إلى Protected و Hidden فسيتم شرحها لاحقا.

ويبين الشكل (١١-٤) شاشة New Property.




**الشكل (١١-٤) شاشة New Property**

ويمكن أيضا إدخال وصف للخاصية في الموقع المحدد description.

ولإضافة method جديدة إلى الصنف الذي تم إنشاؤه نتبع ما يلي:-

من قائمة Class نختار New Method.

في شاشة New Method نطبع اسم الصيغة.

نحدد نوع الصيغة (Public، Protected، Hidden)

نختار مربع Access check box لإنشاء Access method، ونختار Assign check box لإنشاء assign method.Access and Assign methods تتيح للمستخدم معالجة العبارات البرمجية عندما يتم الاستفسار عن قيمة الخاصية أو عند تغيير قيمة هذه الخاصية.

**و . حماية وإخفاء عناصر الصنف Protecting and Hiding Class Members**

الخصائص والصيغ في تعريف الصنف هي بديهيا من نوع Public، وهذا يعني أن العبارات البرمجية في صنوف أخرى أو Procedures معينة يمكنها تغيير خصائص الصنف أو استدعاء صيغه. إما في حالة Protected فإن الخصائص والصيغ يمكن معالجتها فقط من الصيغ الأخرى في تعريف الصنف أو من الصنوف الثانوية التابعة لهذا الصنف. وفي حالة Hidden فإن الخصائص والصيغ يمكن معالجتها من قبل العناصر الأخرى في نفس الصنف فقط والصنوف الثانوية لا يمكن أن تعالج العناصر من نوع Hidden.

# ١٢ . مفاهيم متقدمة Advanced Concepts

نتناول هنا بعض المفاهيم المتقدمة في VFP وبالأخص ربطه مع تطبيقات Windows، والمفهوم الأساسي للتطبيقات التي تسمى بربط قاعدة البيانات المفتوحة (open database connection ODBC) كما ويتم التطرق إلى مفهوم تطبيقات الخادم والعميل (client-server Application).

## أ . إضافة ربط الكائنات واحتضانها (Object Linking and Embedding OLE)

نستطيع التوسع في التطبيقات التي يتم إنشاؤها في VFP من خلال إدراج تطبيقات أخرى ضمن التطبيق المنشأ أو إضافة ما يسمى ActiveX controls. ففي النماذج التي يتم تصميمها أو في الحقول داخل الجداول التي يكون نوع بياناتها عام (general) نستطيع إضافة النصوص، الأصوات، الصور، أو مقاطع الفيديو التي يتم جلبها أو ربطها من التطبيقات الأخرى الموجودة على الحاسبة.

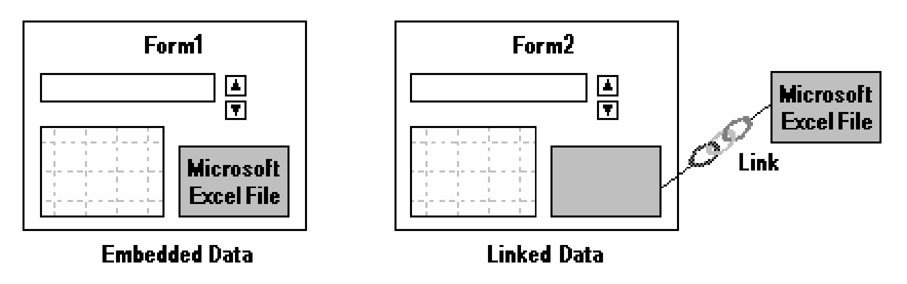
## ب . تصميم تطبيق من نوع OLE

من الممكن أن تكون كائنات الربط هي عميل (clients) بالنسبة إلى VFP، كما ويمكن أن تكون هذه الكائنات هي الخادم (server) إلى VFP. العناصر التي تعمل كخادم تقدم كائنات (objects) إلى بقية العناصر، أما العناصر التي تعمل كعميل فإنها تستطيع إنشاء objects.

من الممكن زيادة كفاءة التطبيق من خلال إدراج كائنات من تطبيقات أخرى مثل (Excel، Word،...الخ).

يمكن ربط الكائن (link) بالتطبيق أو احتضان الكائن (embed) بالنماذج أو بالجداول، فعلى سبيل المثال نستطيع ربط أو احتضان نص من Word في حقل من نوع general في جدول معين، و ربط أو احتضان الكائن في النموذج.

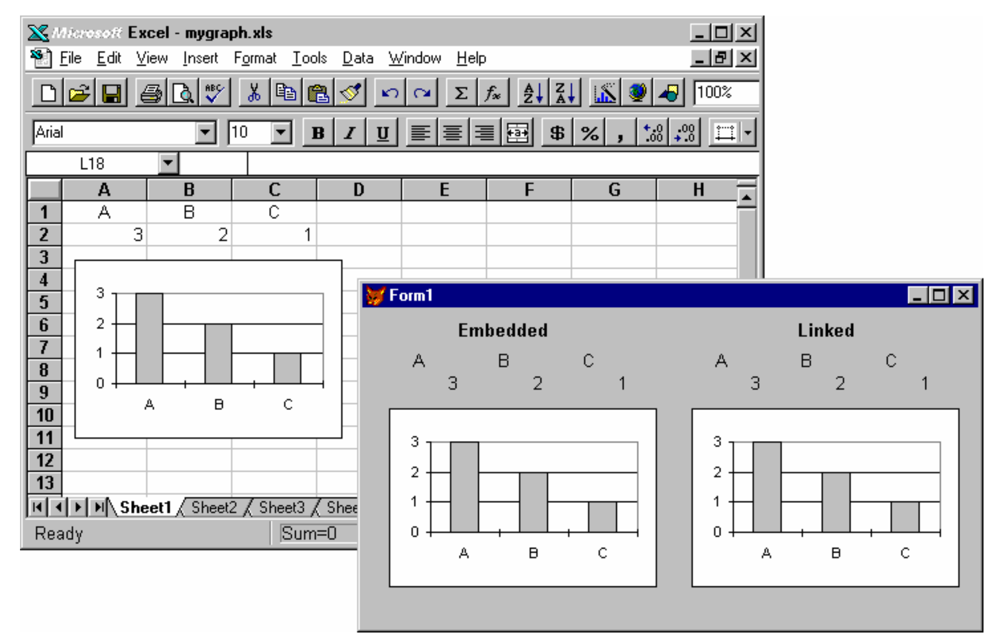
الفرق بين embedding وبين linking يقع في مكان خزن data. فالكائن المحتضن (embedded) تخزن بياناته في الجدول أو النموذج، أما الكائن المربوط (linked) فإن بياناته تبقى في التطبيق الأب، وما يخزن في تطبيق VFP هو فقط عنوان الوصول إليه.




والشكل (١٢-١) يبين رسم توضيحي للكائنات المحتضنة والكائنات المربوطة.

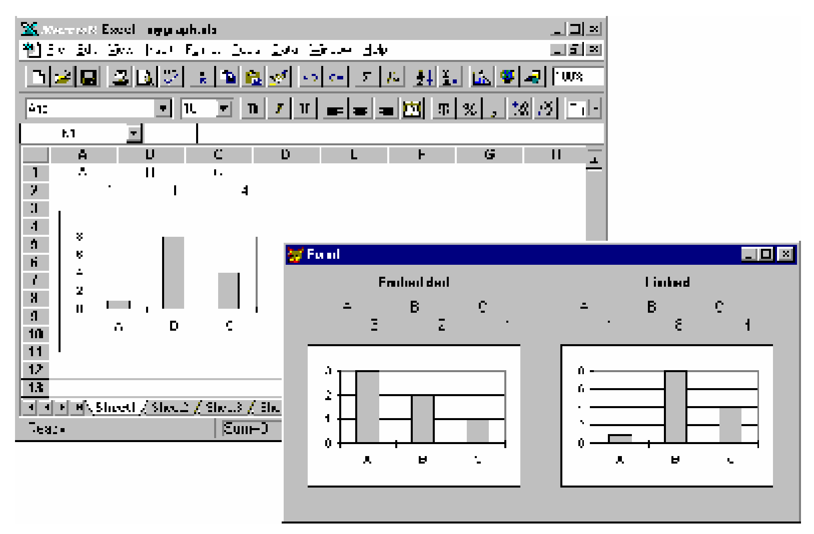
**الشكل (١٢ - ١) مثال على كائن مربوط وكائن محتضن**

ويبين الشكل (١٢-٢) نموذج لورقة عمل من Excel، مربوط ومحتضن ضمن نموذج في VFP اسمه Form، فإذا تم تغيير البيانات في Excel، فإن هذه التغيير سينعكس مباشرة على الكائن داخل النموذج.




**الشكل (١٢-٢) نموذج لكائن مربوط ومحتضن (linked and embedded)**

بينما يوضح الشكل (١٢-٣) نموذج لكائن محتضن فقط، وهذا الكائن هو عبارة عن ورقة عمل من Excel أيـضا. في هذه الحالة إذا تم تغيير بيانات ورقة العمل في Excel فإن التغيير سوف لن ينعكس في النموذج داخل التطبيـق. وهذا المنطق يعكس صحة المفهوم الذي تم شرحه مسبقا وهو أن الكائن المحتضن داخل النموذج يتم خـزن بياناتـه ضمن النموذج (أو في الجدول إذا تم إدراجه في حقل من نوع general)، بينما في حالة الربط سيكون هناك فقط مـا يشير إلى عنوان الكائن في التطبيق الرئيسي، لذلك فإن التغيير في بيانات الكائن في التطبيق الخادم سينعكس مباشـرة في الكائن داخل التطبيق العميل.



**الشكل (١٢-٣) نموذج لكائن محتضن في نموذج**

**ج . إضافة كائن مربوط أو غير مربوط**

في أي نموذج أو تقرير، نستطيع إنشاء objects مرتبطة بحقل من نوع general في جدول معين، مثل هذه الكائنات تسمى bound OLE Objects، ونستخدمها لعرض محتويات OLE objects في حقل general. ولإنشاء bound OLE objects نستخدم المسيطر OLE Bound الموجود في Form Controls Toolbar. ولإنشاء unbounded OLE نستخدم المسيطر OLE Container، ومثل هذا Object لا يرتبط مع general field.

نستطيع إضافة OLE objects إلى الجداول أو النماذج إما بصورة مباشرة أو برمجيا.

**د . إضافة OLE objects إلى الجداول**

عند تصميم جداول قاعدة البيانات وظهور الحاجة إلى إدراج OLE Objects في الجداول، مثلا إذا كان لدينا جدول مواد في المخزن وأردنا إضافة نص Word يحتوي على شرح تفصيلي لكل نادة ومواصفاتها الفنية...الخ. فيجب أولا إضافة general field إلى الجدول ومن ثم إضافة Word document إلى الجدول إما بربطه بالحقل أو احتضانه فيه.

ولإضافة OLE Object إلى جدول نتبع ما يلي:-

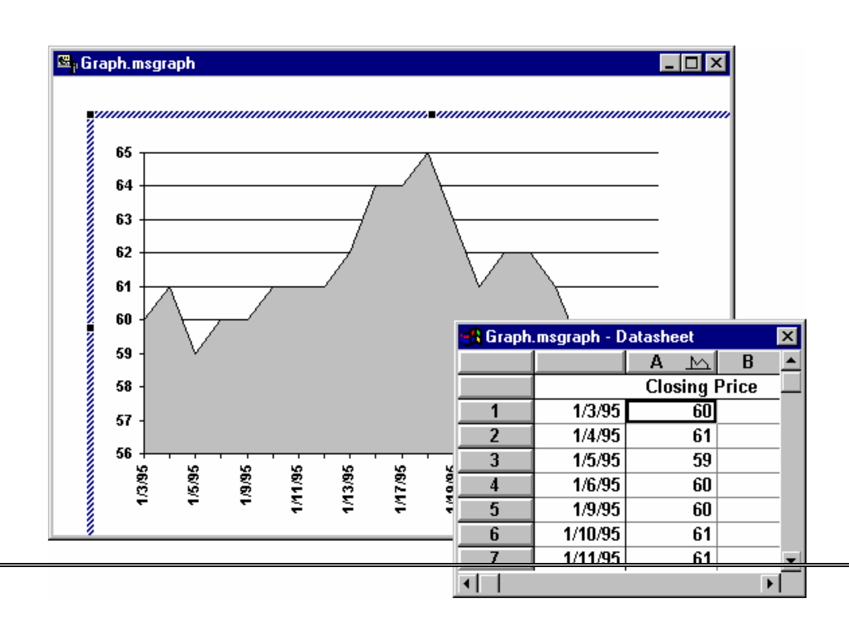
في مصمم الجداول Table Designer ننشأ حقل من نوع general.

نعمل browse إلى الجدول ونعمل Double Click على general field أو إيعاز Modify General.

من قائمة Edit نختار Insert Object (أو نستخدم Append Genral).

**هـ . إعادة تنشيط Microsoft Graph**

Microsoft Graph هو كائن محتضن. والقيم في Microsoft Graph مبنية على أساس القيم في ورقة البيانات فيه (data sheet). كما في الشكل (١٢-٤).



**الشكل (١٢-٤) نموذج من Microsoft Graph**

ولغرض تغيير البيانات برمجيا في Graph، يجب إنشاء String لغرض احتواء البيانات الجديدة، tabs، carriage returns، line feeds، وتعبير هذا string إلى graph من خلال مقطع DATA الموجود في الإيعاز Append General.

ولتوضيح الفكرة نستخدم المثال التالي الذي نفترض فيه وجود جدول اسمه stock، ومن ضمن حقوله حقل التاريخ وحقل سعر الإقفال (آخر سعر). الكائن Microsoft Graph مخزون في الحقل msgraph (من نوع general) في هذا الجدول. هذا المثال يعيد تنشيط graph لبيانات آخر ثلاثين يوما.

| **Code** | **Comments** |
|---|---|
| #DEFINE CRLF CHR(13)+CHR(10)<br>#DEFINE TAB CHR(9)<br>LOCAL lcData | Define carriage return and tab characters. |
| SELECT date, close;<br>FROM Stock WHERE BETWEEN(date, ;<br>DATE(),DATE() - 30) ;<br>ORDER BY date INTO CURSOR wtemp | Select the values that you want to update the graph with, in this case, the date and closing values for stocks for the last 30 days. |
| SELECT wtemp<br>lcData = " " + ;<br>TAB + "Closing Price" + CRLF<br>SCAN<br>lcData = lcData + DTOC(date)<br>lcData = lcData + TAB<br>lcData = lcData + ;<br>ALLTRIM(STR(close)) + CRLF<br>ENDSCAN | Build a character string) `lcData`) of data from the cursor to refresh the graph.<br>“Closing Price ”،as the column header, is the text that will be displayed by default in the graph’s legend. |
| SELECT graph<br>APPEND GENERAL msgraph DATA lcData | Send the new values to the graph in the DATA clause of the APPEND GENERAL command. |
| USE IN wtemp | Close the cursor |

ولإضافة OLE object إلى Form نتبع ما يلي:-

من مصمم النماذج (form designer) نختار OLE Container فيفتح مربع حوار Insert Object.

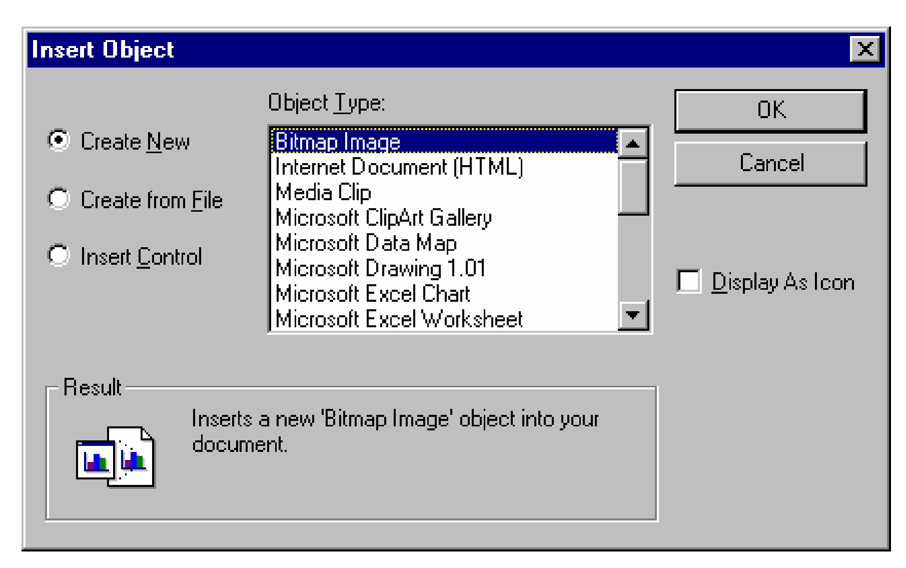
من مربع حوار Insert Object نختار Create New أو Create from file، كما في الشكل (١٢-٥).

نختار OLE object المناسب من قائمة Object List.

ولإضافة OLE Object إلى form controls toolbar نتبع ما يلي:-

من قائمة Tools نختار Options. فتظهر لنا الشاشة كما في الشكل (١٢-٦).

من Controls tab نختار ActiveX controls.

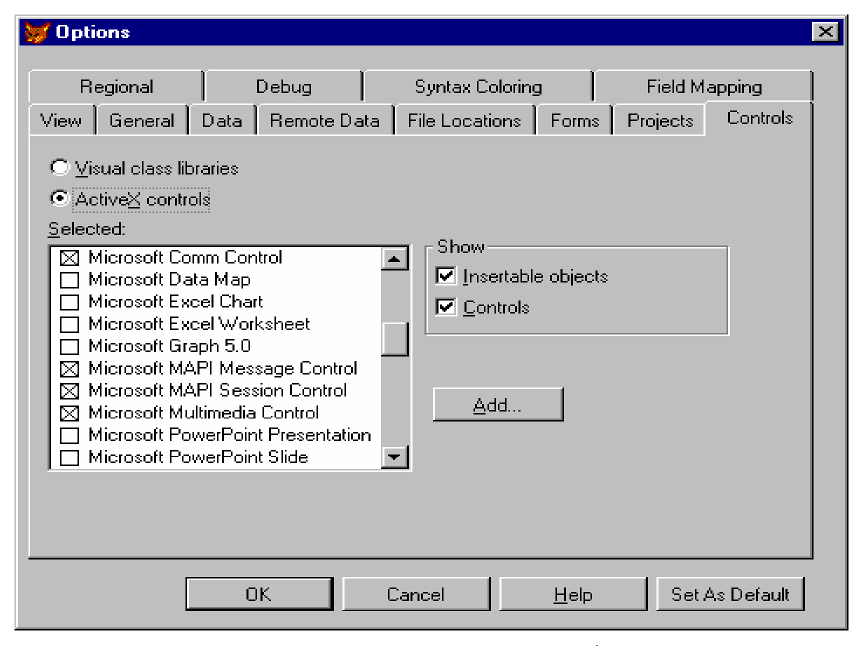



من قائمة Selected نختار OLE object و ActiveX controls المطلوب.

**الشكل (١٢-٥) مربع الحوار Insert Object**

نختار Set as Default ومن ثم OK.

من Form controls toolbar نختار View Classes ومن ثم نختار ActiveX.




**الشكل (١٢-٦) شاشة Options**

ولغرض عرض OLE Object من حقل من نوع General نتبع ما يلي:-

من From Designer نضيف OLE Bound إلى النموذج.

نحدد الحقل من نوع General الذي يحتوي على البيانات عن طريق تغيير الخاصية ControlSource.

فعلى سبيل المثال، إذا كان لدينا جدول اسمه Inventory وحقل General فيه اسمه current، فنقوم بتغيير الخاصية ControlSource إلى Inventory.current.

كما ويمكن عرض OLE Object في النموذج برمجيا وكما يلي:-

| Code | Comments |
|---|---|
| frm1 = CREATEOBJECT("form") | Create form. |
| frm1.ADDOBJECT("olb1", "oleboundcontrol") | Add control. |
| frm1.olb1.ControlSource = "Inventory.Current" | Bind the data to the control. |
| frm1.olb1.Visible = .T.<br>frm1.Visible = .T. | Make the control and form visible. |

## هـ . تصميم تطبيقات العميل/الخادم Client/Server Applications

يوفر VFP أدوات وإمكانيات تصميم تطبيقات client/server قوية وكفوءة. تطبيقات client/server في VFP تمتلك خصائص القوة، السرعة، واجهات المستخدم الرسومية، الاستفسارات المعقدة، التقارير، عمليات النقل النشيطة، وأمنية المعلومات الداخلية.

بناء تطبيق Client/Server عالي الكفاءة يتضمن الاستفادة من سرعة ماكينة VFP. ويتم هذا من خلال تقنيات جديدة مثل SET-BASED DATA ACCESS، و بناء استفسارات قياسية لجلب البيانات التي يحتاجها العميل من الخادم وليس كل القاعدة، وتوزيع جداول قاعدة البيانات بأوفق صورة ممكنة، وموازنة العمل بإجراءات (Procedures) الموجودة عند العميل والإجراءات الموجودة عند الخادم.

وقبل استخدام هذه التقنيات، لا بد من تصميم النظام بشكل يمكن الاستفادة الكاملة من نتائج تطبيق هذه التقنيات. ولا بد من وضع إجابة للتساؤلات التالية:-

أي الجداول سيتم خزنها في server عن بناء التطبيق.

أي الجداول ستكون ذات فائدة أكبر إذا تم خزنها في client بشكل lookup tables.

أي المنظورات (views) سنحتاجها لمعالجة البيانات الموجودة في server.

أي من قواعد المعالجة للتطبيق سيتم تنفيذها من قبل server وكيفية تقاطع العمل بين التطبيق على client مع هذه القواعد.

## و . إنشاء تطبيقات العميل/الخادم Client/Server

يمكن لتطبيق Client/Server معالجة البيانات الموجودة في Server عن طريق:-

المنظورات البعيدة Remote Views.

SQL pass-through.

تعتبر طريقة Remote Views أسهل طريقة لمعالجة وتحديث البيانات على Server. بينما تـوفر طريقـة SQL pass-through (وهي أصعب نسبيا) إمكانية إرسال عبارات SQL من Client إلـى Server مباشـرة، ولكونهـا تعالج في Back-End Server فهي طريقة توفر السرعة والكفاءة في تطبيقات Client/Server، ويوضح الجـدول التالي مقارنة بين طريقتي Remote View و SQL Pass-Through.

| Remote View | SQL Pass-Through |
|---|---|
| Based on a SQL SELECT statement. | Based on any native server SQL statement, enabling data definition statements or execution of server stored procedures. |
| Can be used as data source for controls at design time. | Can't be used as a data source for controls. |
| Provides no ability to execute DDL commands on data source. | Provides method for using DDL commands on data source. |
| Fetches one result set. | Fetches one or multiple result sets. |
| Provides built-in connection management. | Requires explicit connection management. |
| Provides built-in default update information for updates, inserts, and deletes. | Provides no default update information. |
| Provides implicit SQL execution and data fetching. | Provides explicit SQL execution and result fetching control. |
| Provides no transaction handling. | Provides explicit transaction handling. |
| Stores properties persistently in database. | Provides temporary properties for SQL pass-through cursor, based on session properties. |
| Employs asynchronous progressive fetching while executing SQL. | Fully supports programmatic asynchronous fetching. |

**جدول مقارنة بين Remote View و SQL Pass-Through**

## ز . مفهوم DAO و ODBC

كائنات معالجة البيانات (Data Access Objects (DAO)) وربط قواعد البيانات المفتوحـة (Open Database Connectivity (ODBC)) همـا مـن نـوع واجهـات البـرامج التطبيقيـة (Application Programming Interfaces(API))، والتي توفر إمكانية التطبيقات بصورة مستقلة عن أي نظام إدارة قواعد بيانات (DBMS).

يستخدم DAO إمكانيات Microsoft Jet Database Engine لتوفير مجموعة من كائنات معالـــجة البيانــات والتـــي تتــضمن:- Database objects، Tabledef objects، Querydef objects، Recordset objects وغيرها.

ODBC يوفر API لها إمكانية توحيد عدد من أنظمة إدارة قواعد البيانات، من خلال استخدام ODBC Drivers وجعلها تعمل كنظام واحد لإدارة قواعد البيانات. يقوم التطبيق باستخدام هذه API باستدعاء مدير ODBC Driver والذي يقوم بدوره بتعبير الاستدعاء إلى Driver مناسب. ويقوم Driver بدوره بالاتــصال مــع DBMS باستخدام لغة الاستفسار المهيكلة (SQL).